CLEIBSON APARECIDO DE ALMEIDA

CRIAÇÃO DE UM PACOTE PARA VISUALIZAÇÃO GRÁFICA DE DISTRIBUIÇÕES DE DENSIDADE DE PROBABILIDADE UTILIZANDO O SOFTWARE R

Curitiba
Novembro/2006

CLEIBSON APARECIDO DE ALMEIDA

CRIAÇÃO DE UM PACOTE PARA VISUALIZAÇÃO GRÁFICA DE DISTRIBUIÇÕES DE DENSIDADE DE PROBABILIDADE UTILIZANDO O SOFTWARE R

Trabalho de conclusão de curso apresentado para a disciplina CE229 – Laboratório de Estatística II do Curso de Graduação de Bacharel em Estatística, Departamento de Estatística, Setor de Ciências Exatas da Universidade Federal do Paraná, como parte das exigências para obtenção do título de Bacharelado em Estatística.

Orientador: Prof. MSc. Adilson dos Anjos

Curitiba

Novembro/2006

TERMO DE APROVAÇÃO

CLEIBSON APARECIDO DE ALMEIDA

CRIAÇÃO DE UM PACOTE PARA VISUALIZAÇÃO GRÁFICA DE DISTRIBUIÇÕES DE DENSIDADE DE PROBABILIDADE UTILIZANDO O SOFTWARE R

Trabalho de conclusão de curso apresentado na disciplina de Laboratório de Estatística II do Curso de Estatística do Departamento de Estatística, Setor de Ciências Exatas, Universidade Federal do Paraná, aprovado pela seguinte banca examinadora:

____

Prof. *MSc.* Adilson dos Anjos - Orientador
Universidade Federal do Paraná

____

Prof. *PhD.* Bruno Grimaldo Martinho Churata
Universidade Federal do Paraná

Curitiba, 29 de novembro de 2006.

“Em minha carreira eu errei mais de 9000 arremessos, perdi mais de 300 partidas e falhei 26 vezes o arremesso decisivo da partida que me fora confiado. Ao longo da vida e da carreira eu errei, errei e errei. Nunca neguei. Por isso fui bem sucedido.”

Michael Jordam

SUMÁRIO

Lista de Tabelas vii
Lista de Figuras vii
RESUMO ix
1 INTRODUÇÃO 1
2 MATERIAIS E MÉTODOS 3
2.1 Recursos Computacionais 3
2.1.1 Computadores 3
2.1.2 Sistema Operacional 5
2.1.3 Versão do R 5
2.1.4 Comparações entre as configurações 6
2.2 Criação de pacotes no R 6
2.2.1 Composição estrutural do pacote 7
2.2.1.1 Descrição dos arquivos raiz 8
2.2.1.2 Descrição dos diretórios 9
2.2.2 Criação de pacotes com Windows 9
2.2.2.1 Windows - Pré-requisitos 10
2.2.2.2 Windows – Instalação e configuração dos Pré-requisitos 11
2.2.2.3 Windows – Criando um pacote 20
2.2.3 Criação de pacotes com Linux 29
2.2.3.1 Linux – Criando um pacote 30
2.3 Recursos do TCLTK 33
2.3.1 TCLTK no R-2.4.0 34
2.3.2 Exemplos práticos 36
3 APLICAÇÃO 38
3.1 O pacote PLOTT 38
3.1.1 As funções gerais do pacote plott 39
3.1.1.1 A função plotti 40
3.1.1.2 A função tabli 41
3.1.2 As funções específicas do pacote plott 43
3.1.2.1 A distribuição Normal 43
3.1.2.2 A distribuição t-Student 46
3.1.2.3 A distribuição X 48
3.1.2.4 A distribuição F 50

3.1.2.5 A distribuição Exponencial 52
3.1.2.6 A distribuição Gama 54
4 CONSIDERAÇÕES FINAIS 57
6 REFERÊNCIAS BIBLIOGRÁFICAS 58
7 ANEXOS 60
7.1 Exemplo de arquivo .Rd 60
7.2 Cd-rom do pacote plott 61
7.3 Código fonte do pacote plott 62

LISTA DE TABELAS

Tabela 1 – Recursos Computacionais 3
Tabela 2 – Comparação entre Sistemas operacionais x pré-requisitos 6
Tabela 3 – Principais funções do tcltk 34
Tabela 4 – Opções da função tabli() 42

LISTA DE FIGURAS

Figura 1 – Diretórios de um pacote 8
Figura 2 – Diretório C:\tools 12
Figura 3 – Diretório C:\Perl 12
Figura 4 – Diretório C:\mingw 13
Figura 5 – Diretório C:\miktex 14
Figura 6 – Diretório C:\html_compiler 15
Figura 7 – Path do windows 16
Figura 8 – Área de trabalho do DOS 17
Figura 9 – Exemplo de verificação 18
Figura 10 – Exibição do path 18
Figura 11 – Correção do Arquivo Vars.pm 19
Figura 12 – Correção da linha 68 (Vars.pm) 20
Figura 13 – Diretório criado: meupacote 23
Figura 14 – Arquivos do diretório meupacote 24
Figura 15 – Editando o DESCRIPTION 24
Figura 16 – Diretório MAN 25
Figura 17 – Prompt do DOS 26
Figura 18 – Compilação do pacote 27
Figura 19 – Instalação do pacote 28
Figura 20 – Diretório src/library após finalização do pacote 29
Figura 21 – Terminal do Linux 31
Figura 22 – Diretório home/cleibson/Desktop após finalização do pacote 33
Figura 23 – Áreas exibidas no painel gráfico 40
Figura 24 – Pacote plott (função plotti) 41
Figura 25 – Pacote plott (função tabli) 42

CRIAÇÃO DE UM PACOTE PARA VISUALIZAÇÃO GRÁFICA DE DISTRIBUIÇÕES DENSIDADE DE PROBABILIDADE UTILIZANDO O SOFTWARE R

Novembro/2006

CLEIBSON APARECIDO DE ALMEIDA

RESUMO

Este trabalho apresenta um conjunto de regras para criação de pacotes utilizando o software R em conjunto com a biblioteca TCLTK. Os pacotes que utilizam recursos do tcltk tem por objetivo facilitar o uso da estrutura base do R criando uma interface de uso amigável e também interagir com o usuário ao ponto de facilitar a atividade realizada naquele instante. Aparentemente podemos seguir as regras sugeridas pelo site R-Project, porém algumas dicas estão ocultas nos manuais disponíveis na internet e estão vinculadas preferencialmente ao sistema operacional Linux. O pacote criado é adicional ao R e tem como finalidade exibir graficamente algumas distribuições de probabilidade (Normal, t-Student, F, Qui-quadrado, Exponencial e Gama). Com esta nova funcionalidade os estudantes de probabilidade poderão simular visualmente o comportamento probabilístico dessas funções e observar seu comportamento após a modificação dos parâmetros iniciais oferecido pelo pacote.

Palavras-chave: Pacotes no R, tcltk, Distribuições Densidade de Probabilidade.

1 INTRODUÇÃO

Atualmente existem várias opções de pacotes para o software R. Alguns, no entanto, tornam o R mais acessível facilitando a utilização e evitando o uso de inúmeras linhas de comando para a realização de determinadas tarefas.

Os pacotes são particularmente objetos providos de carregamento adicional ao R e são geralmente caracterizados por um conjunto de funções capazes de automatizar determinadas tarefas. Durante a instalação inicial do R o usuário insere automaticamente alguns pacotes adicionais, considerados de uso geral (R DEVELOPMENT CORE TEAM, 2005).

Para a correta utilização de um pacote é necessária a leitura preliminar de seu manual de utilização para o entendimento restritivo daquele objeto que será carregado no R. Também é necessário atenção para o uso correto do pacote, pois os pacotes adicionais são geralmente utilizados para casos especiais de análise estatística.

Ao criador de pacotes são necessários alguns conhecimentos específicos como lógica de programação, assunto relativo à aplicação do pacote e também dominar as funções primordiais do R. Atualmente o R possui vários pacotes prontos para serem utilizados e outros sendo desenvolvidos pelos mais diversos pesquisadores em torno do mundo, mas nem todos os problemas estão resolvidos pois existem uma infinidade de aplicações que precisam de pacotes para serem automatizadas pelo R.

O Tcl (Tool Command Language) é uma linguagem de script usada por meio milhão de programadores ao redor do mundo e se tornou um componente importante em milhares de corporações. Com uma sintaxe simples de programação pode ser usado como um utilitário desktop ou embutida em outros programas como será sua aplicação neste trabalho. Tk é um toolkit para criação de interface gráficas com o usuário, possibilitando assim criar GUIs[1] (Guide User Interface) poderosas e inacreditavelmente rápidas, ao contrário de aplicações desenvolvidas em outras linguagens interpretadas ou até mesmo pré compiladas (MONTEIRO, 2004).

Tanto o Tcl como o Tk foram desenvolvidos por John Ousterhout[2]. O Tcltk é altamente portátil e pode ser executado em praticamente todas as plataformas

---

[1] GUI: Guide User Interface são ambientes visuais exibidos pelo computador.

[2] John Ousterhout: Pesquisador Britânico que desenvolveu os recursos do tcltk.

computacionais como Unix (Linux, Solaris, IRIX, AIX, BSD) , Windows e Macintosh (FRANCA, 2005).

O software R possui um pacote próprio chamado **tcltk** e para executá-lo dentro do ambiente basta ao usuário executar o comando de carregamento de pacotes. Este pacote tem como propósito fornecer recursos gráficos provenientes de ações internas do R tais como, janelas, menus, botões e recursos gráficos comumente vistos em programas computacionais (DALGAARD, 2001).

Neste trabalho será criado um pacote adicional ao R que terá por finalidade exibir graficamente algumas distribuições de probabilidade (Normal, t-Student, Qui-Quadrado, F, Exponencial e Gama). Com esta nova funcionalidade os cursistas e interessados em probabilidade poderão simular visualmente o comportamento probabilístico dessas funções e observar seu comportamento após a modificação dos parâmetros iniciais oferecido pelo pacote.

2 MATERIAIS E MÉTODOS

2.1 Recursos Computacionais

Neste tópico serão abordadas as características de hardware[3], sistemas operacionais, software[4] e versão do R utilizada para a criação dos pacotes. Também será feita uma análise comparativa indicando facilidades e dificuldades de cada configuração.

Quando se trata de desenvolvimento de aplicações uma questão fundamental é saber quais são os dispositivos de hardware que aquela aplicação necessita para funcionar adequadamente. Os recursos de hardware atuais foram projetados para satisfazer uma gama de aplicações, inclusive os recursos do `tcltk`.

Em seguida, devemos nos ater aos recursos de software. Destacamos que o desenvolvimento de pacotes no R só será possível se tivermos em mãos três recursos de software: sintaxe, linguagem e bibliotecas compatíveis ao R conforme mostra a tabela 1 (R MAILING LISTS, 2005).

Tabela 1 – Recursos Computacionais

Recurso de hardware/software	hardware/software
Dispositivo de entrada	Mouse e teclado
Dispositivo de saída	Monitor
Dispositivos centrais	Placa mãe, processador e memória
Sintaxe de Programação	`PERL`
Linguagem dos Manuais de Ajuda	`HTML` e `TEX`
Biblitecas de Compilação	`MINGW (windows)` ou `GCC (linux)`
Sintaxe de Programação	`PERL`

2.1.1 Computadores

Um dos fatores que podem influenciar no desenvolvimento de uma aplicação computacional é o computador utilizado. É interessante testar a aplicação em vários computadores com diferenças entre hardware, software e sistema operacional,

[3] Hardware: É a parte física do computador, ou seja, é o conjunto de componentes eletrônicos, circuitos integrados e placas, que se comunicam através de barramentos computacionais.

[4] Software: Também conhecido como programa de computador.

assim teremos maior convicção de que a aplicação desenvolvida funcionará em multiplataformas. Neste trabalho serão utilizados três computadores com diferentes configurações de hardware e objetivos no trabalho:

Computador 1:

- Placa mãe: MSI K8N NEO
- Processador: AMD ATHLON 64 PROCESSOR
- Memória: DDR 512 mb
- Placa de Vídeo: RADEON 9600
- Dispositivos de entrada: TECLADO GENIUS USB ABNT2, MOUSE USB ÓPTICO
- Dispositivos de saída: MONITOR SANSUMG LCD
- Sistema Operacional: WINDOWS XP SERVICE PACK 2
- Versão do R instalada: 2.4.0
- Objetivo: Criar pacote para windows

Computador 2:

- Placa mãe: PC CHIPS P31G
- Processador: INTEL CORE 2 EXTREME
- Memória: DDR 1024 mb
- Placa de Vídeo: ON BOARD
- Dispositivos de entrada: TECLADO PS2 ABNT2, MOUSE PS2
- Dispositivos de saída: MONITOR SANSUMG 793V
- Sistema Operacional: SUSE LINUX 9.3
- Versão do R instalada: 2.4.0
- Objetivo: Criar pacote para Linux

Computador 3:

- Placa mãe: INTEL DESKTOP BOARD D975XBX2
- Processador: INTEL CORE 2 DUO
- Memória: DDR 512 mb
- Placa de Vídeo: NVIDIA GFORCE 5500
- Dispositivos de entrada: TECLADO PS2 ABNT2, MOUSE PS2 ÓPTICO

- Dispositivos de saída: MONITOR SANSUMG 510N
- Sistema Operacional Dual-boot: WINDOWS 2000 e UBUNTU LINUX 6.06
- Versão do R instalada: 2.4.0
- Objetivo: Testar pacotes criados

2.1.2 Sistema Operacional

Foram utilizados quatro diferentes sistemas operacionais em duas plataformas computacionais sendo classificadas em:

- **Plataforma Linux:**
  - SUSE[5] LINUX 9.3
  - UBUNTU[6] LINUX 6.06

- **Plataforma Windows:**
  - Microsoft Windows XP service pack 2
  - Microsoft Windows 2000

2.1.3 Versão do R

A versão utilizada para a criação do pacote foi a 2.4.0, porém os testes foram realizados nas versões 2.3.1 e 2.4.0.

Embora não haja diferenças relevantes entre as duas versões testadas, é importante realizar o teste para que não tenhamos problemas de incompatibilidade entre as versões. O pacote não foi testado em versões antigas, pois algumas dependências sugeridas pelo pacote criado não funcionam em versões anteriores à 2.0.

---

[5] Suse: Disponível em http://www.novell.com/linux/

[6] Ubuntu: Disponível em http://www.ubuntu.com/

2.1.4 Comparações entre as configurações

Embora inicialmente não seja necessário fazer comparações entre as configurações dos computadores utilizados vale lembrar que o não seguimento dos pré-requisitos iniciais podem comprometer o desenvolvimento do trabalho, tais como:

- A versão do `R-2.4.0` é estável em determinado hardware e sistema operacional?
- A apresentação dos recursos `tcltk` são exibidos adequadamente no sistema operacional?
- O sistema operacional tem suporte a desenvolvimento de pacotes no R?
- Existe falta ou limitação de recursos gráficos em determinado sistema operacional computacional?

Tabela 2 – Comparação entre Sistemas operacionais x pré-requisitos

	Estabilidade do R-2.4.0	Tcltk é exibido corretamente?	Suporte desenvolvimento	Limitação gráfica
Windows 2000	estável	sim	não	não
Windows XP	estável	sim	não	não
Ubuntu Linux	estável	não	sim	sim
Suse Linux	estável	sim	sim	não

De acordo com a tabela 2 podemos verificar alguns problemas iniciais em relação às características de sistemas operacionais e os recursos necessários para a criação dos pacotes no R (R MAILING LISTS, 2005).

2.2 Criação de pacotes no R

A criação de pacotes é feita com utilização dos recursos oferecidos pelo `R-2.4.0`. Independentemente da plataforma utilizada, os comandos que geram o pacote são os mesmos, porém, as configurações de armazenamento do pacote no computador são diferentes em cada plataforma computacional.

Para abordar estas diferenças temos neste trabalho o processo de criação do pacote em duas plataformas computacionais diferentes.

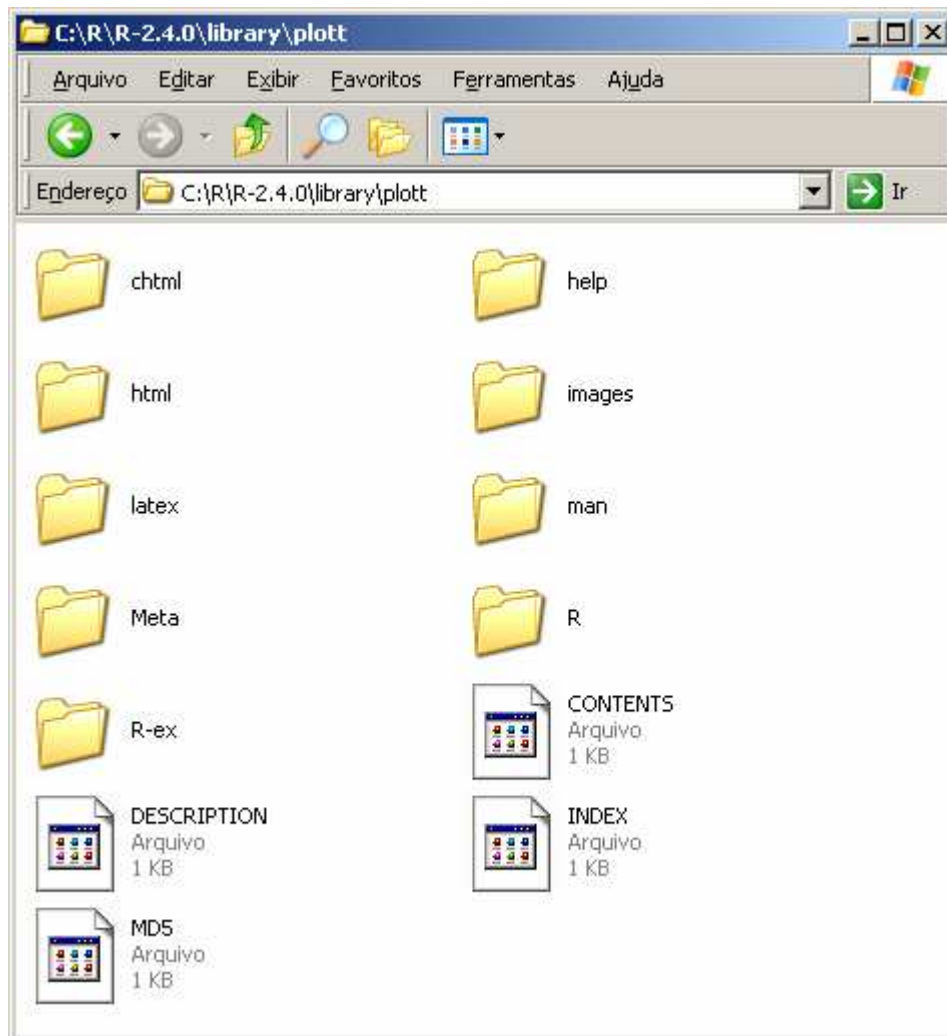
Primeiramente será abordada a utilização da plataforma Windows ao qual são utilizados recursos do DOS que, por sua vez, é a base deste sistema operacional. Para criar pacotes com o Windows é necessária uma série de configurações no sistema, dentre eles a emulação do ambiente UNIX, o servidor PERL, compilador MINGW e o interpretador da linguagem TEX.

Em seguida temos a abordagem sobre a criação do pacote com utilização da plataforma Linux, pois esta plataforma, apesar de complexa, trata com facilidade os recursos ligados a linguagem de programação PERL e, também, possui em sua base diversos compiladores dentre eles o GCC, o qual é necessário para compilar o pacote para que possa ser instalado futuramente.

Com esta breve introdução já podemos observar que existem algumas divergências entre as plataformas computacionais e também as configurações do sistema operacional para a criação dos pacotes.

2.2.1 Composição estrutural do pacote

Os pacotes do R seguem um padrão de diretórios. Esse padrão foi definido pela equipe de desenvolvedores do R e tem por finalidade deixar os pacotes com as mesmas características. Isso faz com que os erros do pacote estejam ligados a ele e não ao R. A figura 1 ilustra os diretórios e arquivos que fazem parte da estrutura interna de um pacote.

Figura 1 – Diretórios de um pacote

2.2.1.1 Descrição dos arquivos raiz

Os **arquivos raiz** do pacote são providos de arquivos de texto sem extensão computacional que tem a finalidade de informar ao R as características de informação, indexação e criptografia do pacote. A seguir estão descritas as funcionalidades de cada arquivo raiz do pacote.

CONTENTS – Este arquivo é responsável pela indexação e interpretação dos arquivos de help e manuais do pacote.

DESCRIPTION – Principal arquivo do pacote, nele estão contidas todas as informações que definem utilidade, autor, data e descrição básica preliminar. Este arquivo pode ser lido mesmo que o pacote não esteja carregado na seção do R.

INDEX – Faz a indexação das funções existente no pacote.

MD5 – Arquivo que criptografa todos os arquivos e diretórios do pacote. A criptografia é fundamental para definir a plataforma computacional e versão do R que o pacote poderá funcionar.

2.2.1.2 Descrição dos diretórios

CHTML – Arquivos de help gerados pela compilação de html. Utilizam extensão .chm.

HELP – Arquivos de help gerados pela compilação do Latex. Não possuem extensão, pois são interpretados diretamente e internamente pelo R.

HTML – Arquivos de help em formato pré-compilado com extensão .html.

IMAGES – É um diretório opcional. Contém arquivos de imagens que são carregadas pelo pacote, utilizam três tipos de extensão: png, jpg e gif.

LATEX – Arquivos de help em formato pré-compilado com extensão .tex.

MAN – Arquivos de documentação gerados após a compilação do pacote. Utiliza a extensão .gz e é um arquivo compactado de arquivos com extensão .Rd gerados antes da compilação do pacote.

META – Arquivos de indexação do pacote. São utilizados quando se busca alguma função respectiva ao pacote em questão e tem extensão .Rds.

R – Este é o diretório mais importante do pacote. Nele estão contidas as funções que fazem o pacote funcionar. Não possuem extensão, pois são interpretados diretamente e internamente pelo R.

R-EX – Arquivos de exemplos. Geralmente está vazio porque os exemplos quase sempre são escritos em arquivos de help.

DATA – É um diretório opcional ao pacote e nele podem conter bases de dados a serem utilizados durante algum procedimento na utilização do pacote.

2.2.2 Criação de pacotes com Windows

A criação de pacotes utilizando o Windows requer uma série de pré-requisitos que se não forem seguidos torna impossível a manipulação dos comandos que irão gerar o pacote.

2.2.2.1 Windows - Pré-requisitos

Os pré-requisitos listados abaixo são necessários para construir um pacote no Windows. Veja a descrição e faça download[7] do pré-requisito para seu computador:

- **TOOLS –** O R foi desenvolvido como ferramenta de desenvolvimento em ambiente Unix, porém para desenvolver aplicações do R em ambientes Windows é necessário emular um ambiente Unix. A ferramenta TOOLS simplesmente faz esta emulação, (FRASCATI, 2006).
  Endereço: *http://www.murdoch-sutherland.com/Rtools/tools.zip*

- **ACTIVE PERL –** Este aplicativo tem por finalidade interpretar recomendações escritas na sintaxe `perl`. O `perl` será a sintaxe do pacote que iremos construir e ele somente funcionará se este pré-requisito estiver instalado corretamente, (FRASCATI, 2006).
  Endereço: *http://www.activestate.com/Products/ActivePerl/Download.html*

- **MINGW –** Para compilar aplicativos escritos em `perl` é necessário que tenhamos um compilador que entenda esta linguagem. Para isso será necessário instalar e configurar corretamente o `mingw`, (ROSSI, 2005).
  Endereço: *http://prdownloads.sf.net/mingw/MinGW-3.2.0-rc-3.exe?download*

- **MIKTEX –** Os arquivos de ajuda escritos na linguagem `tex` precisam ser compilados para serem exibidos. Para compilar os arquivos de ajuda escritos em `tex` será necessário instalar e configurar o **miktex**,

[7] Download – Copiar um arquivo disponível na internet para um computador local. Também usa-se o termo **baixar** da internet.

(FRASCATI, 2006).
Endereço: *http://www.miktex.org/*

- **HTML HELP COMPILER -** Os arquivos de ajuda escritos na linguagem `html` precisam ser compilados para ser exibidos. Para compilar os arquivos de ajuda escritos em `html` será necessário instalar e configurar o **html help compiler**, (FRASCATI, 2006).
  Endereço:
  http://msdn.microsoft.com/library/default.asp?url=/library/enus/htmlhelp/html/hwmicrosofthtmlhelpdownloads.asp

- **WINZIP –** É um compactador de arquivos. Este aplicativo será necessário para finalizar o pacote uma vez que a extensão dos pacotes para windows é `.zip`, (FRASCATI, 2006).
  Endereço: *http://www.winzip.com/*

2.2.2.2 Windows – Instalação e configuração dos Pré-requisitos

Tendo em vista que todos os pré-requisitos já foram baixados da internet para o computador e estão prontos para serem instalados, vamos, passo-a-passo fazer a instalação de cada um deles.

Antes de mais nada devemos ficar atentos ao local onde serão instalados e como padronização neste trabalho vamos fazer todas as instalações em `C:/`. Instalar em `C:/` irá facilitar o trabalho de configuração do `path`, que será descrito mais adiante.

TOOLS – Extrair o conteúdo do arquivo **tools.zip** para o diretório **C:/tools** de forma que o diretório fique como a figura 2.

Figura 2 – Diretório C:\tools

ACTIVE PERL – Instale o **Active Perl** no diretório **C:/Perl** de forma que o diretório fique como a figura 3.

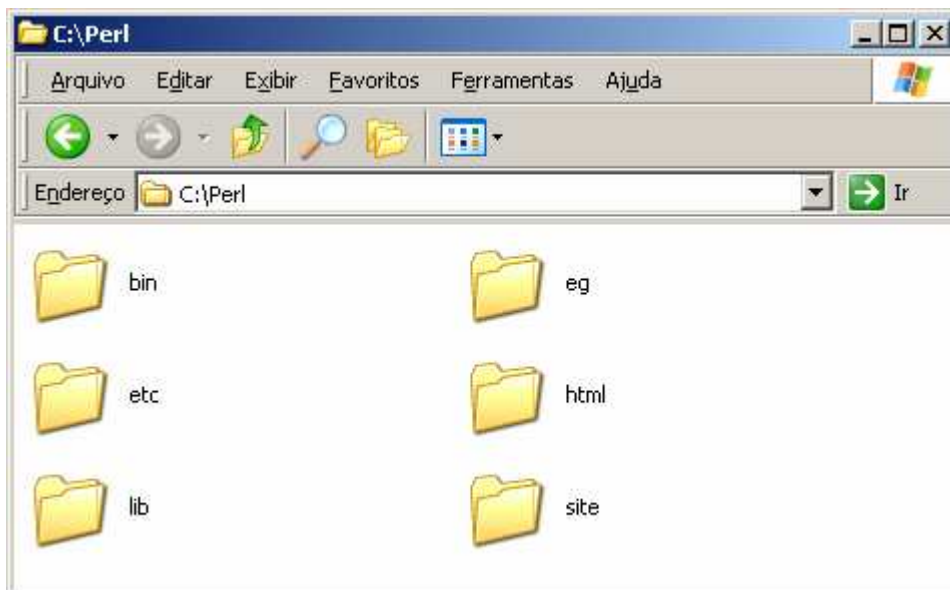
Figura 3 – Diretório C:\Perl

MINGW - Instale o **Mingw** no diretório **C:/mingw** de forma que o diretório fique como a figura 4.

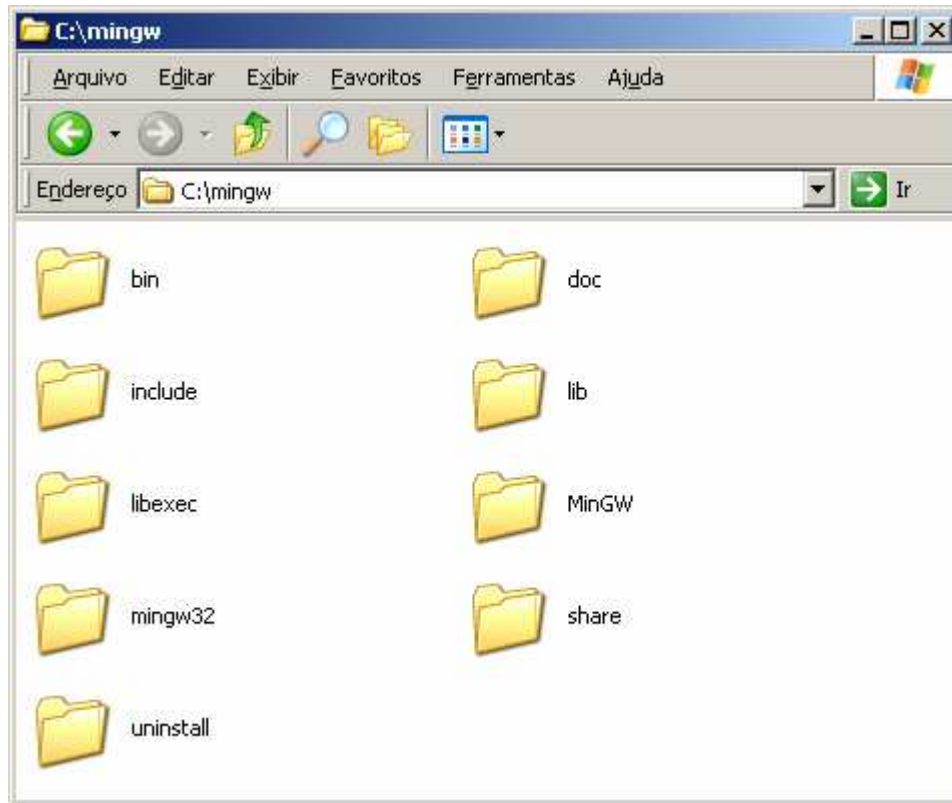
Figura 4 – Diretório C:\mingw

MIKTEX - Instale o **Miktex** no diretório **C:/miktex** de forma que o diretório fique como a figura 5.

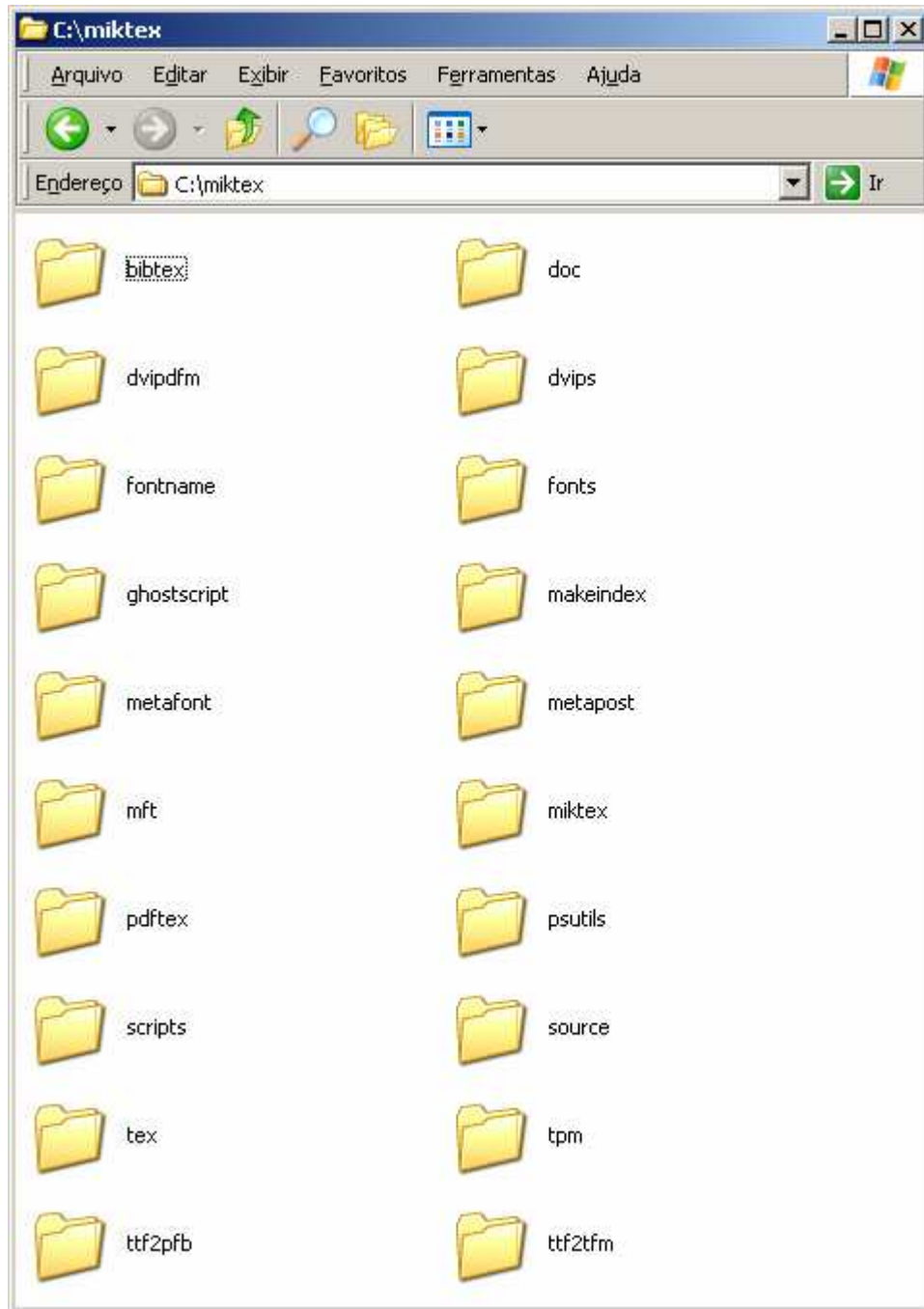
Figura 5 – Diretório C:\miktex

HTML HELP COMPILER - Instale o **Html help compiler** no diretório **C:/html_compiler** de forma que o diretório fique como a figura 6.

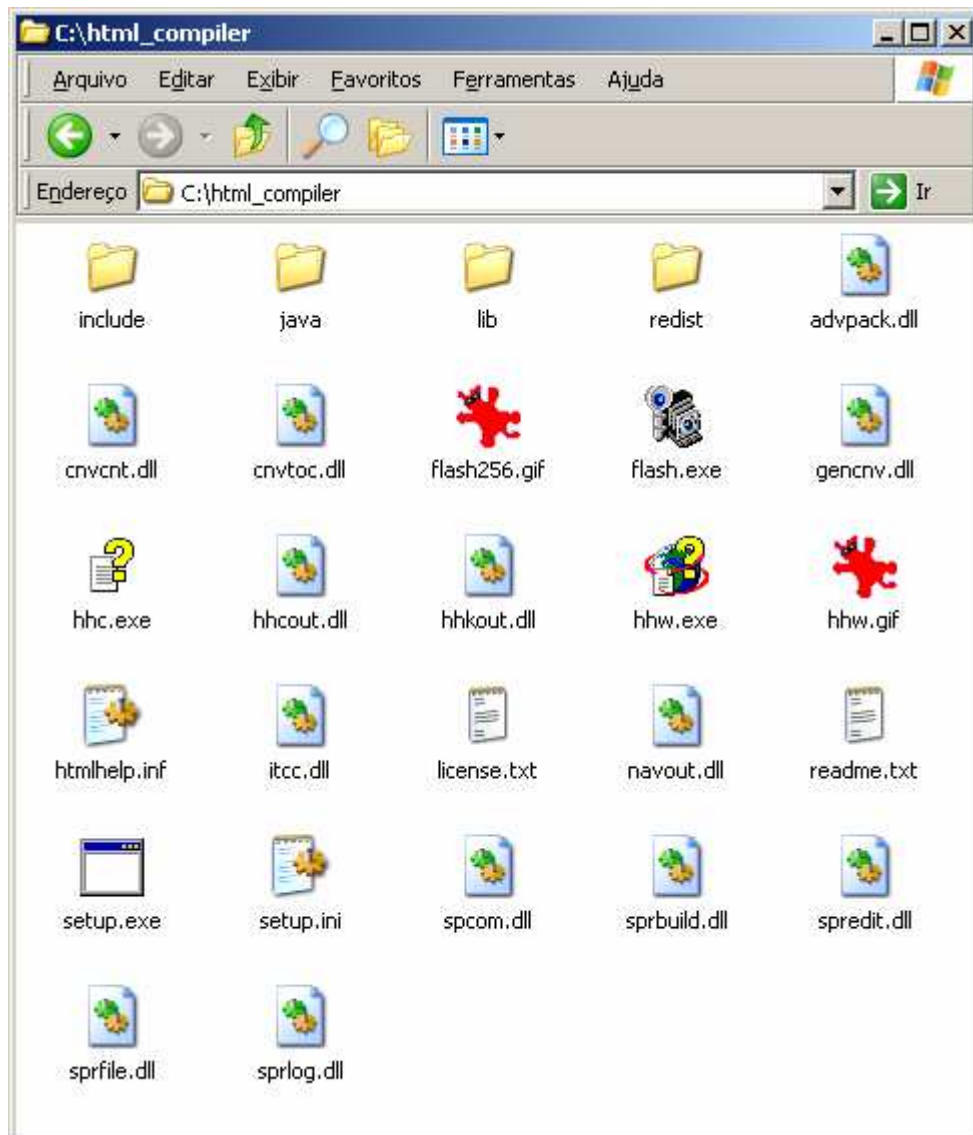
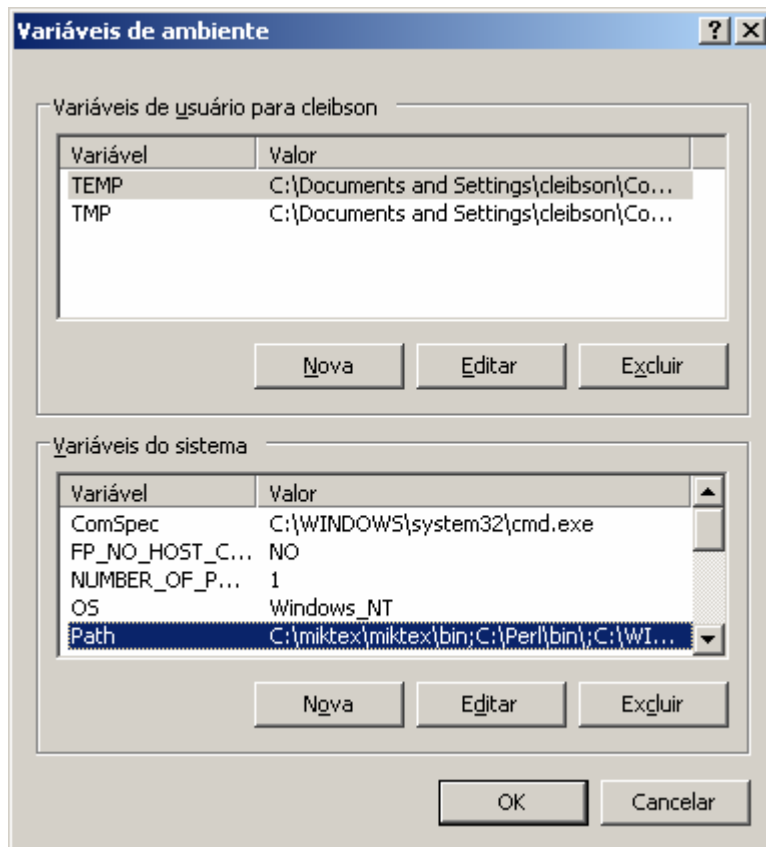
Figura 6 – Diretório C:\html_compiler

Pronto. Agora que todos os pré-requisitos estão instalados será necessário configurá-los para que funcionem adequadamente em conjunto ao sistema operacional. Para isso siga o caminho em `INICIAR/PAINEL DE CONTROLE/SISTEMA/AVANÇADO/ VARIÁVEIS DE AMBIENTE` e edite o `path` do windows conforme mostra a figura 7.

Figura 7 – Path do windows

Adicione ao path as seguintes linhas:

```
C:\miktex\miktex\bin;
C:\Perl\bin\;
C:\tools\bin;
C:\html_compiler;
C:\mingw\bin;
C:\R\R-2.4.0\bin; (desde que o diretório bin do R esteja nesta seqüência)
```

Por último basta verificar se todos os pré-requisitos estão instalados. Para isso vamos em *INICIAR/EXECUTAR* e digitamos *cmd*. Perceba que irá abrir uma janela do DOS (figura 8).

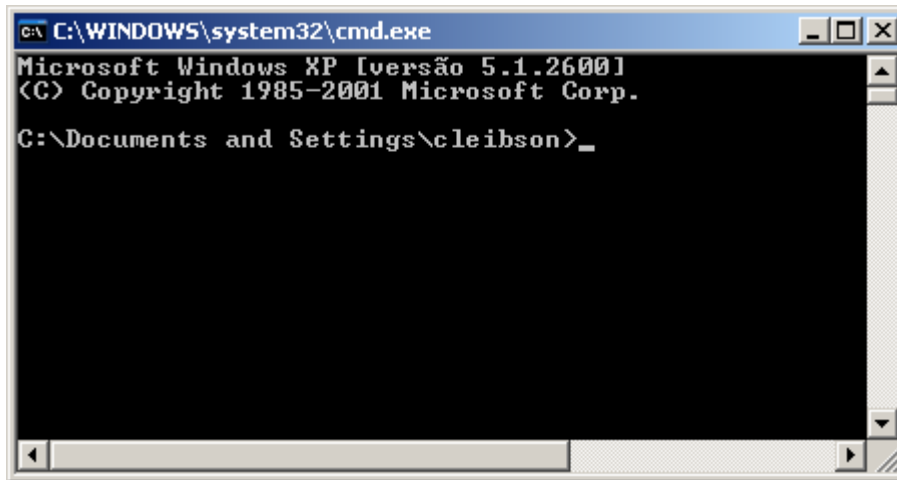
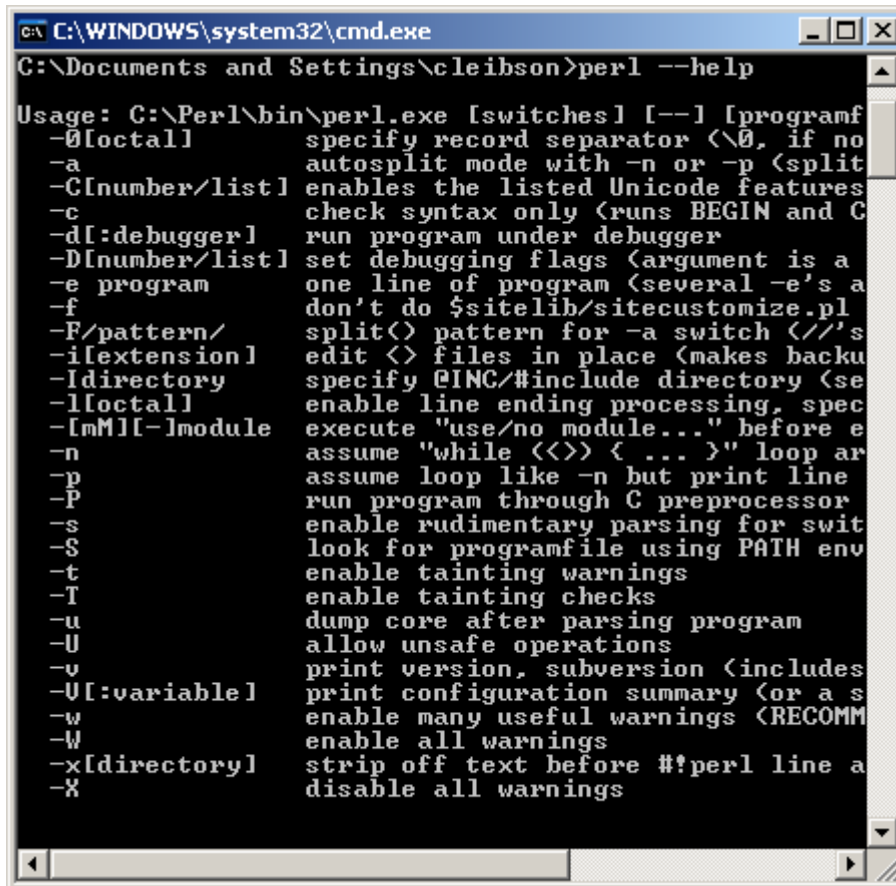
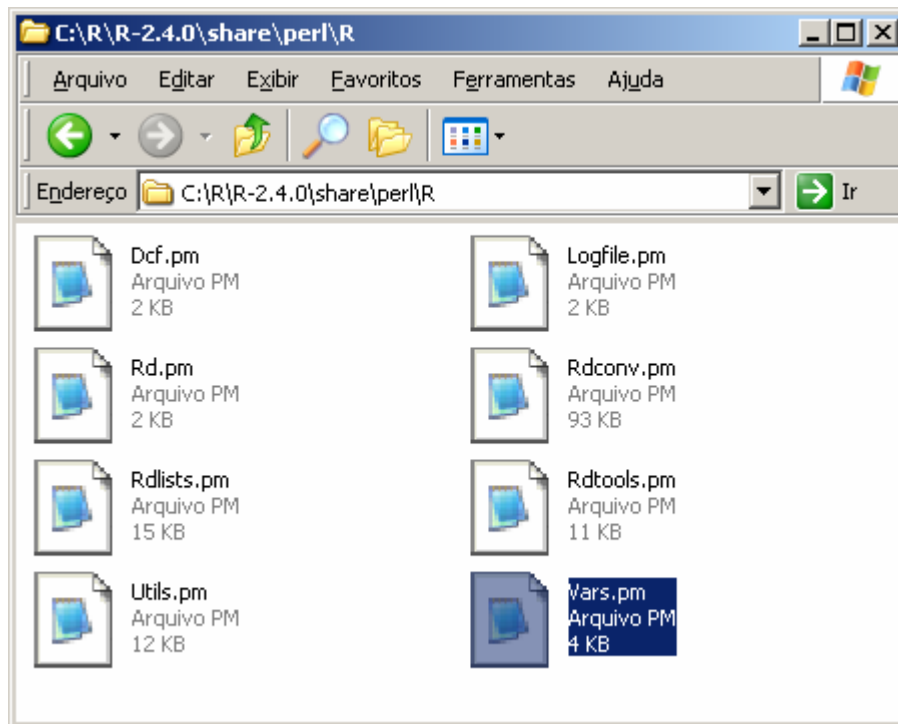
Figura 8 – Área de trabalho do DOS

Nesta janela, faça a verificação dos pré-requisitos digitando os comandos abaixo, veja figura 9, (MURDOCH, 2005).

```
Perl -help
Dos2unix -help
Tex -help
Gcc -help
R
```

Figura 9 – Exemplo de verificação

Para ter certeza de que todos os pré-requisitos estão funcionando observe se nos cinco casos o terminal deu alguma resposta exibindo os caracteres de ajuda do comando digitado. No caso do R, observe que abrirá o `R-2.4.0` em ambiente DOS. Caso não tenha dado algum retorno verifique no `path`, se o caminho está correto. Para verificar todos os path's digite o comando *`path`* (figura 10).

Figura 10 – Exibição do path

Após a instalação, configuração e verificação dos pré-requisitos temos que corrigir um pequeno **erro** no R. Vá no diretório raiz do R e siga o caminho `R\share\perl\R` e procure pelo arquivo **Vars.pm** (figura 11), abra esse arquivo em um editor de texto.

Figura 11 – Correção do Arquivo Vars.pm

Após abrir o arquivo, procure a linha 68 e troque:

`$TMPDIR = "_"` por `$TMPDIR = "$R_HOME/temp"` conforme figura 12.

Figura 12 – Correção da linha 68 (Vars.pm)

```
61      $R_EXE = "Rterm.exe";
62      $R_CMD = "Rcmd.exe";
63      getenv("TMPDIR", "TMPDIR", "C:/TEMP");
64      if (-d $TMPDIR) {
65      $TMPDIR = Win32::GetShortPathName($TMPDIR) if
66      $TMPDIR =~ s+\\+/+g;  ## ensure forward slash
67      } else {
68      $TMPDIR = "$R_HOME/temp"
69      }
70  }
71  else{
72      if($R_HOME){
73      $R_EXE = "${R_HOME}/bin/R";
74      }
```

Agora entre no diretório raiz do R e crie um diretório chamado *temp*. Em resumo esta técnica corrige um erro que ao construir pacotes o R irá solicitar um diretório temporário para a criação prévia do pacote.

2.2.2.3 Windows – Criando um pacote

Durante a criação de pacotes em ambiente Windows iremos utilizar apenas duas ferramentas.

- **R-2.4.0 –** Será responsável por gerar o pacote básico, ou seja, criará o pacote pré-compilação. Este pacote deverá ser editado e as instruções estarão num arquivo denominado **leia-me e apague-me**. Basicamente as modificações estarão ligadas aos manuais e arquivos de help.

- **Prompt do DOS –** Será responsável pela emulação do ambiente Unix no Windows, compilação e finalização do pacote.

Passo 1 – Criando funções no R.

Criar funções no R é uma tarefa bem metódica. Basta seguir a sintaxe abaixo:

```
Variável <- function (parâmetros da função) { desenvolvimento da sintaxe}
```

Exemplo 1: Somando dois números.

```
soma<-function (a,b){
            somaab<-a+b
            return(somaab)}
#Para executar a função use:
#soma(coloque o primeiro número, coloque o segundo número)
Exemplo: soma(3,4)
```

Exemplo 2: Elevando um número ao quadrado.

```
aquadrado<-function(a){
            aq<-a^2
            return(aq)}
#Para executar a função use:
#aquadrado(coloque o número)
Exemplo: aquadrado(3)
```

Para saber mais sobre como escrever funções no R leia atentamente o **capítulo 10 - Writing your own functions** (R DEVELOPMENT CORE TEAM, 2005).

Passo 2 – Utilizando a função `package.skeleton`

A função *“`package.skeleton`”* automatiza a criação de um novo pacote. Cria toda a estrutura dos diretórios base e salva as funções e dados que irão compor o pacote. Além do mais cria todos os arquivos de help e recursos estruturados do pacote. Porém, deve-se editar os arquivos `.Rd` para que realmente sejam o help do pacote em questão.

Como foi descrito anteriormente o *`package.skeleton`* cria um modelo do arquivo de help no formato `.Rd` e sua edição pode ser feita utilizando o bloco de notas ou outro editor de texto.

A sintaxe do `package.skeleton` é dada por:

```
package.skeleton(name = "anRpackage", list, environment = .GlobalEnv,
path = ".", force = FALSE)
```

onde os argumentos:

- **name:** Deverá ser uma string (texto) e será o nome do novo pacote.
- **list:** Deverá ser um vetor listando os objetos (funções) do R para colocar no pacote. Deverão ser listadas apenas as funções que irão fazer parte do pacote.
- **environment:** Se o argumento “list” for omitido, os índices deste ambiente serão empacotados, ou seja, todas as funções criadas na atual seção do R irão compor o pacote.
- **path:** Caminho para colocar os diretórios dentro do pacote.
- **force:** Se for “FALSE” não sobrescreverá um pacote já existente.

Passo 3 – Criação e empacotamento de funções

Primeiramente crie um caminho onde será salvo o pacote, dê preferência ao diretório *SRC/LIBRARY* onde é permitido a criação de diretórios. Pode ser que outros caminhos não tenham permissão de gravação pelo windows. Fique atento ao local onde está instalado o R em seu computador.

Então:

```
caminho<-c("C:\\R\\R-2.4.0\\src\\library")
```

Agora vamos criar quatro funções, elas vão fazer a soma, multiplicação, divisão e subtração entre dois números.

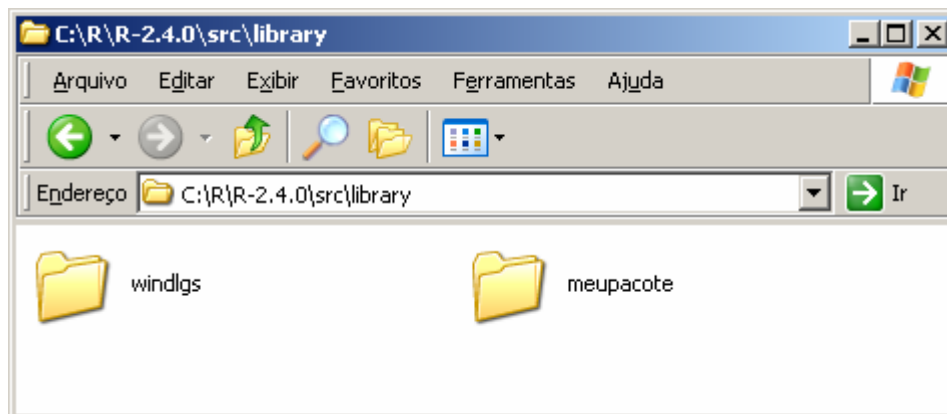
```
soma<-function (a,b){
            calculo<-a+b
            return(calculo)}
multi<-function (a,b){
            calculo <-a*b
            return(calculo)}
divisi<-function (a,b){
            calculo <-a/b
```

```
            return(calculo)}
subtra<-function (a,b){
            calculo <-a-b
            return(calculo)}
```

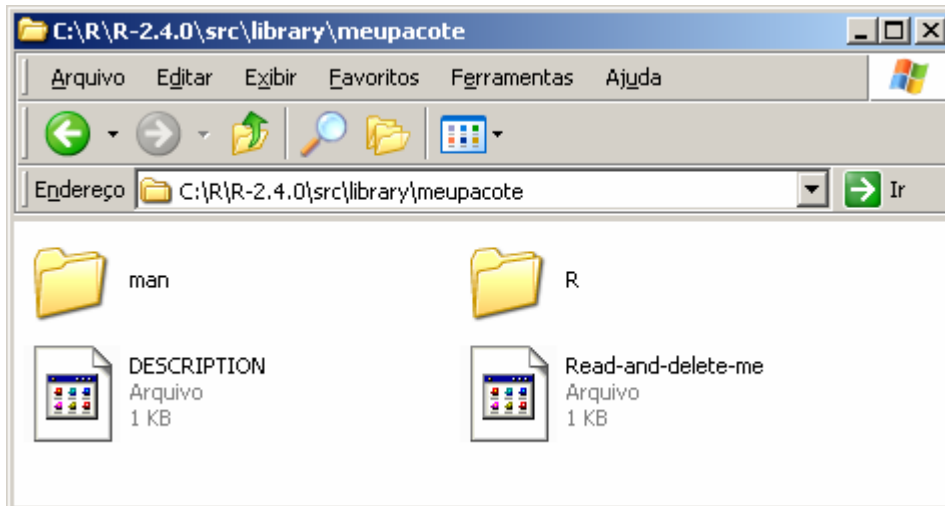
Por último vamos executar a função package.skeleton.

```
      package.skeleton(list=c("soma","multi","divisi","subtra"),
name="meupacote", path=caminho)
```

Perceba que foi criado um diretório chamado **meupacote** no caminho sugerido (figura 13).

Figura 13 – Diretório criado: meupacote

Passo 4 – Editando os arquivos de manuais e ajuda

Perceba que dentro do diretório **meupacote** (figura 14) foram criados dois diretórios e 2 arquivos na raiz. Caso tenha dúvida sobre estes diretórios e arquivos leia o item 2.2.1.1 e 2.2.1.2 deste trabalho.

Figura 14 – Arquivos do diretório meupacote

Leia atentamente o arquivo **Read-and-delete-me** utilizando um editor de texto, perceba que ele nos dá algumas instruções para a correta finalização do pacote. Após a leitura apague este arquivo.

Agora, abra o arquivo **DESCRIPTION** com um editor de texto e edite conforme figura 15. Em seguida salve o arquivo, (CAREY, 2002).

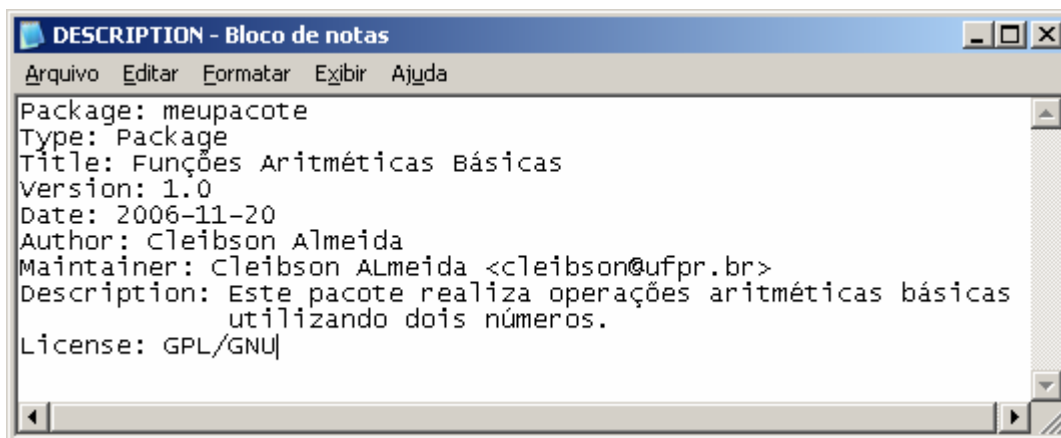
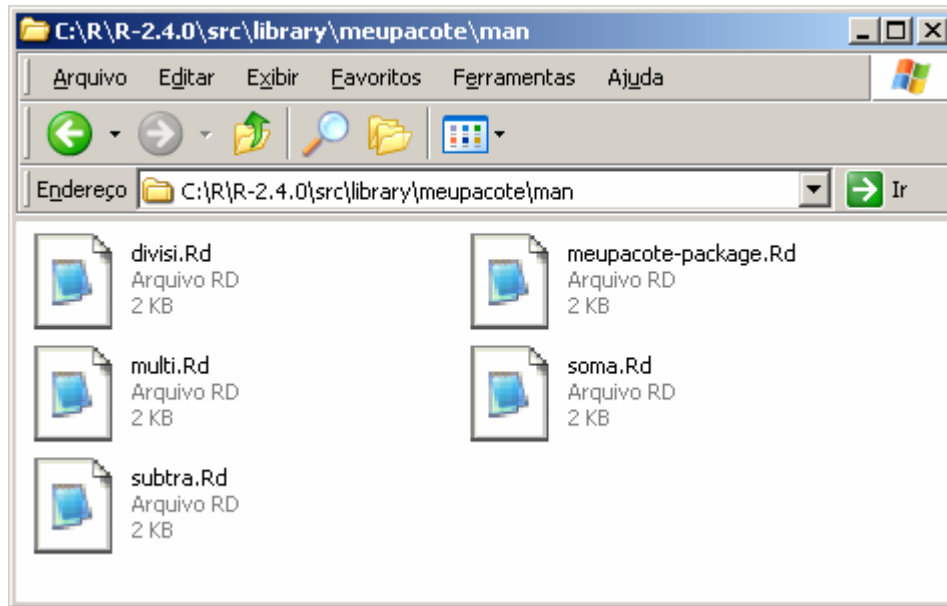
Figura 15 – Editando o DESCRIPTION

Como informado pelo **Read-and-delete-me** edite os arquivos do diretório MAN (figura 16).

Figura 16 – Diretório MAN

Para facilitar a edição veja um modelo de arquivo `.Rd` no anexo 7.1 deste trabalho. Pronto, o empacotamento foi realizado e finalizado.

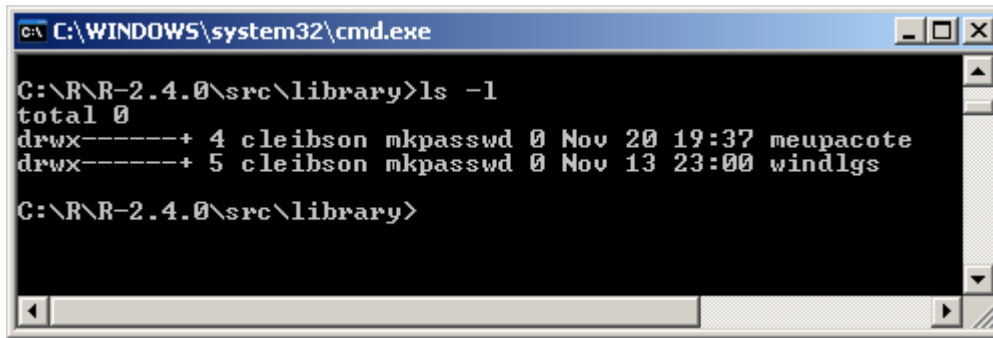
Passo 5 – Compilação do Pacote

Após a edição de todos os arquivos necessários devemos sair do R e utilizar o prompt do DOS. Portanto siga o caminho `INICIAR/EXECUTAR` e digite *cmd*.

Em posse do DOS navegamos até o local onde está nosso pacote (figura 17). Para isso utilize os comandos abaixo:

Para listar o conteúdo de um diretório use: *`ls -l`*

Para avançar utilize: *`cd <nome do diretório >`*

Exemplo*:* *`cd windows/temp`*

Para retornar utilize: *`cd ..`*

Para retornar dois diretórios utilize: *`cd ../..`*

Para retornar três diretórios utilize: *`cd ../../..`*

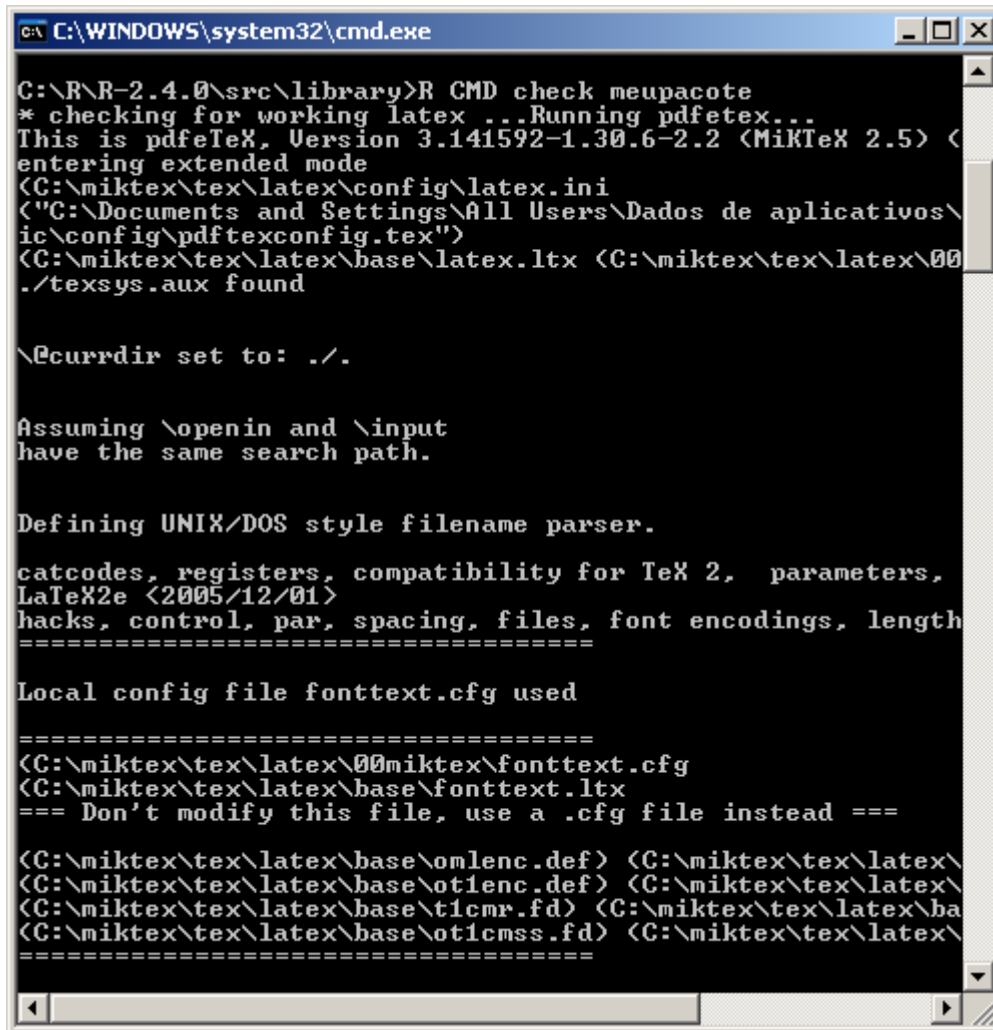
E assim por diante até que você fique no diretório onde foi salvo o pacote.

Figura 17 – Prompt do DOS

```
C:\WINDOWS\system32\cmd.exe

C:\R\R-2.4.0\src\library>ls -l
total 0
drwx------+ 4 cleibson mkpasswd 0 Nov 20 19:37 meupacote
drwx------+ 5 cleibson mkpasswd 0 Nov 13 23:00 windlgs

C:\R\R-2.4.0\src\library>
```

Para compilar o pacote utiliza-se o comando:

```
R CMD check <nome do pacote>
```

Em nosso caso:

```
R CMD check meupacote
```

Perceba que este comando irá checar e criar cada componente do pacote final (figura 18). Em caso de erros o pacote não será compilado.

Figura 18 – Compilação do pacote

Agora que já temos o pacote compilado só precisamos instalar e verificar o funcionamento do pacote no R.

Passo 6 – Instalação do pacote

Podemos instalar o pacote através do prompt do DOS utilizando o comando (figura 19).

```
R CMD INSTALL <nome do pacote>
```

Em nosso caso:

```
R CMD INSTALL meupacote
```

Figura 19 – Instalação do pacote

```
C:\WINDOWS\system32\cmd.exe

C:\R\R-2.4.0\src\library>R CMD INSTALL meupacote

---------- Making package meupacote ------------
  adding build stamp to DESCRIPTION
  installing R files
  installing man source files
  installing indices
  installing help
Warning: \alias{soma} already in divisi.Rd -- skipping the one in meupacote-pack
age.Rd
Warning: \alias{soma} already in divisi.Rd -- skipping the one in multi.Rd
Warning: \alias{soma} already in divisi.Rd -- skipping the one in soma.Rd
Warning: \alias{soma} already in divisi.Rd -- skipping the one in subtra.Rd
 >>> Building/Updating help pages for package 'meupacote'
     Formats: text html latex example chm
  divisi                            text    html    latex   example
  meupacote-package                 text    html    latex   example
  multi                             text    html    latex   example
  soma                              text    html    latex   example
  subtra                            text    html    latex   example
  adding MD5 sums

* DONE (meupacote)

C:\R\R-2.4.0\src\library>_
```

Passo 7 – Finalizando o pacote

Para finalizar o pacote utilizamos o comando:

```
R CMD build <nome do pacote> (para salvar o pacote binário)
```

Em nosso caso:

```
R CMD build meupacote
```

E para criar uma versão no formato .zip a ser instalada no R versão windows, basta utilizar o comando:

```
R CMD build --binary --use-zip <nome do pacote>
```

Em nosso caso:

```
R CMD build --binary --use-zip meupacote
```

Passo 8 – Extras

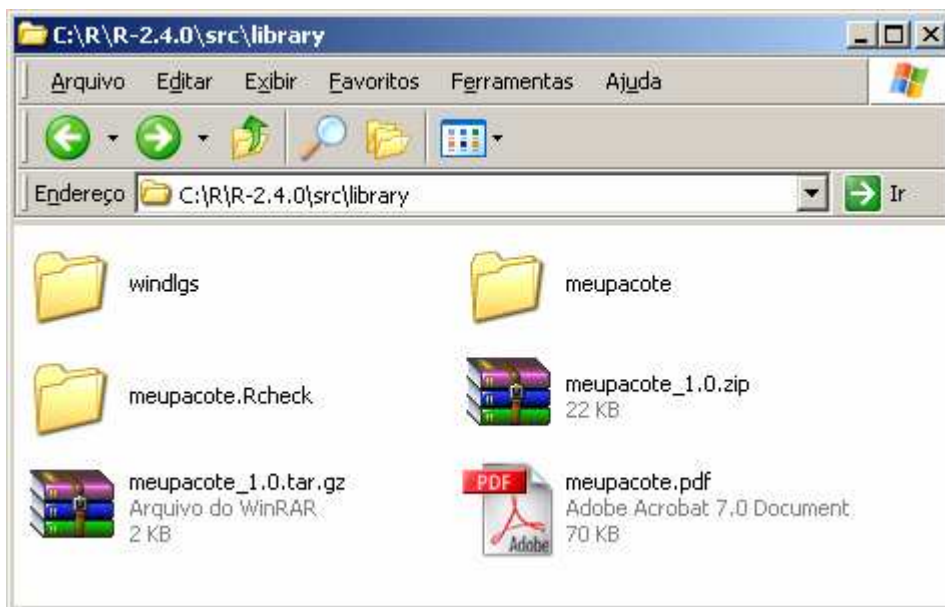
Caso queira criar um manual de ajuda em .pdf basta utilizar o comando:

```
R CMD Rd2dvi --pdf <nome do pacote>
```

Em nosso caso:

```
R CMD Rd2dvi --pdf meupacote
```

Por fim teremos um diretório `C:/R/R-2.4.0/SRC/LIBRARY` com os seguintes arquivos (figura 20):

Figura 20 – Diretório src/library após finalização do pacote

- meupacote – diretório onde está o pacote pré-compilação.
- meupacote.Rcheck – diretório onde está o pacote após compilação.
- meupacote_1.0.zip – pacote pronto para ser instalado no windows.
- meupacote_1.0.tar.gz – pacote em formato binário.
- meupacote.pdf – manual do pacote no formato .pdf.

2.2.3 Criação de pacotes com Linux

A criação de pacotes utilizando o Linux requer apenas a instalação da biblioteca GCC. Esta biblioteca contém a maioria dos plugins[8] e compiladores dos softwares produzidos em ambientes Unix, dentre eles estão os compiladores de pacotes do R.

2.2.3.1 Linux – Criando um pacote

Criar pacotes no linux é muito mais fácil do que se imagina, pois todos os recursos e componentes necessários para compilação são instalados por default no Linux. Caso não tenha o `GCC` instalado e atualizado, procure o site responsável pela distribuição que você está utilizando e veja como instalar e atualizar a biblioteca `GCC`.

No geral, se preocupe com o local onde serão salvos os arquivos do pacote e seus diretórios, geralmente são salvos em `/home/usuário` quando não é definido um `path` na função `package.skeleton`. Na criação de pacotes para Linux vamos utilizar apenas a janela do terminal[9] conforme mostra a figura 21.

Logo, esteja logado como `ROOT`[10], pois podem ocorrer casos em que você não tenha permissão para gravação de arquivos.

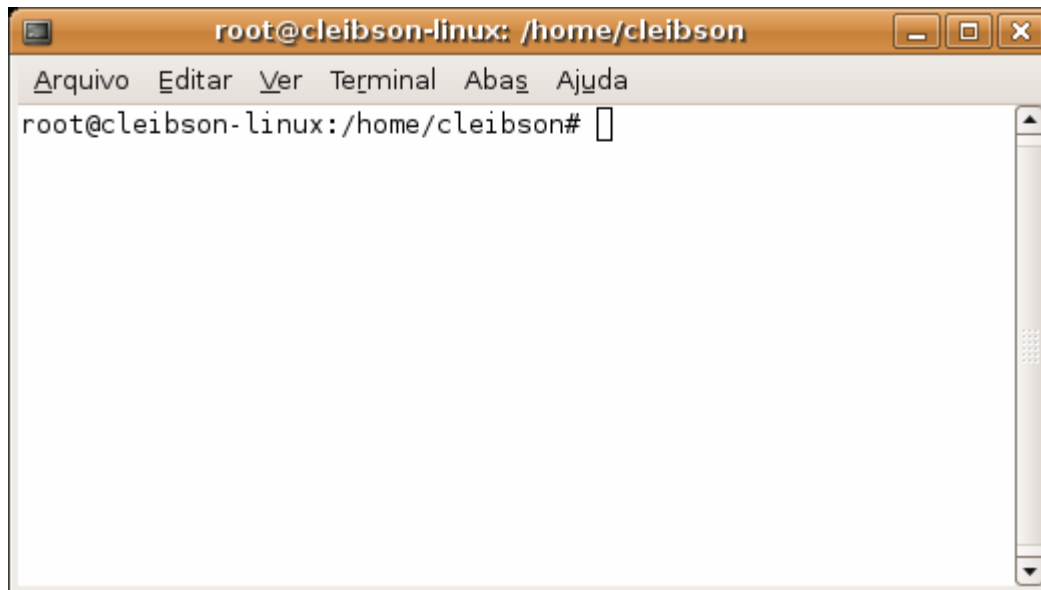
- **Terminal** – Será responsável pela emulação do R, compilação e finalização do pacote.

---

[8] Plugins: São programas de computador que servem normalmente para adicionar funções a outros programas para prover alguma função particular.

[9] Terminal: Emulador e gerenciador linux em modo texto.

[10] Root: É o usuário linux que tem o poder total sobre as atividades do sistema, podendo ler, executar e gravar qualquer ação.

Figura 21 – Terminal do Linux

Passo 1 – Criando funções no R.

Vide passo 1 no item 2.2.2.3.

Passo 2 – Utilizando o comando `package.skeleton`

Vide passo 2 no item 2.2.2.3.

Passo 3 – Criação e empacotamento de funções

Primeiramente abra o R digitando o comando `R.` Crie um caminho onde será salvo o pacote, dê preferência ao diretório `home/nome do usuário/Desktop` onde fica mais fácil visualizar o que está sendo feito. Neste caso:

```
caminho<-c("/home/cleibson/Desktop")
```

Após definido o caminho siga as orientações do passo 3 no item 2.2.2.3.

Passo 4 – Editando os arquivos de manuais e ajuda

Vide passo 4 no item 2.2.2.3.

Passo 5 – Compilação do Pacote

Vide passo 5 no item 2.2.2.3.

Passo 6 – Instalação do pacote

Vide passo 6 no item 2.2.2.3.

Passo 7 – Finalização do pacote

Vide passo 7 no item 2.2.2.3.

Nota: Repare que foi utilizado o argumento - - use-zip para gerar pacotes em tar.gz, isso deve-se ao fato que o linux tem com padrão compactar seus arquivos neste formato.

Passo 8 – Extras

Por fim teremos um diretório *`home/cleibson/Desktop`* com os seguintes arquivos (figura 22):

Figura 22 – Diretório home/cleibson/Desktop após finalização do pacote

- meupacote – diretório onde está o pacote pré-compilação.
- meupacote.Rcheck – diretório onde está o pacote após compilação.
- meupacote_1.0_R_x86_64-pc-linux-gnu.tar.gz – pacote pronto para ser instalado no Linux.
- meupacote_1.0.tar.gz – pacote em formato binário (pré-compilado).

2.3 Recursos do TCLTK

Hoje em dia um dos grandes problemas enfrentados pelos desenvolvedores de soluções é a diversidade de plataformas computacionais que poderão ser encontradas nas instituições. O Tcltk é chamado de programação de aplicações para integração visual pois funciona praticamente em qualquer plataforma e sistema operacional.

Durante o desenvolvimento de aplicações, a plataforma de integração está ficando tão estrategicamente importante quanto o sistema operacional e os bancos de dados. Tcltk é a melhor plataforma de integração por causa de sua velocidade de uso, amplitude de funcionalidade, e características para empreendimento prontas como linha-segurança, internacionalização e desenvolvimento multiplataforma (FRANCA, 2005).

2.3.1 TCLTK no R-2.4.0

O R por sua vez possui total suporte ao tcltk tendo em sua base um pacote exclusivo para o desenvolvimento de recursos tcltk integrados aos recursos funcionais do programa. Porém vale lembrar que os ambientes visuais são gerados em tk e o servidor para que os ambientes funcionem são chamados de tcl (DALGAARD, 2002).

Para carregar o pacote tcltk no R, digite o comando:

```
library(tcltk)
```

Este pacote (tcltk) possui em sua base 235 funções, que também são chamadas de widgets, das quais neste trabalho serão exploradas as funções mais usuais, aproximadamente quinze. Os itens da tabela 3 mostram os detalhes das funções mais usuais do tcltk.

Tabela 3 – Principais funções do tcltk

Descrição da Função	**Sintaxe**	**Exemplo**
Tktoplevel: Esta função cria uma janela vazia onde possam ser criados objetos de integração. Também é chamada de widget mãe e todos os demais widgets dependem de sua criação para poderem funcionar adequadamente.	*tktoplevel()*	tt <-tktoplevel()
Tklabel: Esta função cria uma etiqueta mostrando informações de texto no plano de fundo da aplicação. É geralmente utilizada para	*tklabel(widget_mãe, text="Texto_ no _plano_de_fundo")*	tklabel(tt, text="Olá mundo!")

mostrar textos que não estão ligados à alguma widget.		
Tkpack: Esta função faz a atualização quando forem adicionadas novas widgets dentro da widget mãe. Pode ser utilizada cada vez que criamos uma nova widget ou podemos aplicá-la após a criação de várias widgets.	*tkpack(nova_widget)*	tkpack(lb1)
Tkbutton: Esta função cria botões para interagir com outras widgets.	*tkbutton(widget_mãe, text=”Texto_do_botão”)*	tkbutton(tt, text=”clique aqui”)
Tktitle: Esta função define um nome para a janela criada. Este nome é exibido na barra superior da janela.	*tktitle(widget_mãe)*	tktitle(tt) <-“Minha Janela”
Tkdestroy: Esta função destrói uma widget mãe existente, ou seja, finaliza uma aplicação tcltk no R.	*tkdestroy(widget_mãe)*	tkdestroy(tt)
Tkentry: Esta função cria uma entrada de texto para a comunicação de widgets	*tkentry(widget_mãe, width=”Largura_do_ campo_de_texto”)*	tkentry(tt, width=30)
Tkgrid: Esta função tem por finalidade organizar e alinhar as widgets existentes dentro da aplicação.	*tkgrid(widget_1, widget_2, ...)*	tkgrid(lb1, nome)
.configure: Esta função tem por finalidade	*<nome_da_widget>.configure (widget_1,*	*tkgrid.configure(lb1, nome, sticky=”w”)*

concluir e exibir a organização e alinhamento gerado lógicamente por qualquer widget.	*widget_2_que_deve _estar_alinhado _com _widget_1, ..., stick= ”nome_d e_um_identificador”)*	tklabel.configure(lb1, nome)
Tkconfigure: Esta função configura as widgets para comunicarem entre si.	*tkconfigure(widget _de_comunicação, text=”Texto _de_retorno_ao_usuário”)*	tkconfigure(nome, text=”Digite um texto”)
Tktext: Esta função nomeia uma widget para que seja indexada em uma ação realizada por outra widget.	*tktext(widget_mãe)*	tktext(tt)
Tkscrollbar: Esta função cria uma barra de rolagem vertical na janela criada com tktoplevel.	*tkscrollbar(widget _mãe,command= ”commando _que_ executará_a _barra_de_rolagem”)*	tkscrollbar(tt, command=function(abr e_rolagem))
Tkradiobutton: Esta função cria um botão de seleção tipo radio. Este tipo de botão é utilizado como opção de escolha entre diversas opções disponíveis.	*tkadiobutton(widget_mãe , text=”Texto_da_seleção”)*	tkradiobutton(tt, text="selecione aqui")
Tkmessage: Esta função exibe uma mensagem de alerta ao usuário.	*tkmessage(widget_mãe, text=”Texto_da_alerta”)*	tkmessage(tt, text="Por favor, clique aqui!")
Tkcheckbutton: Esta função cria caixas de seleção para fazer comunicação entre widgets.	*tkcheckbutton (widget_mãe, text=”Texto_da_opção”)*	tkcheckbutton (tt, text="Cheque esta opção”)

2.3.2 Exemplos práticos

No item 2.3.1 foram vistos algumas definições e exemplos teóricos com o uso do tcltk. Neste item serão abordados dois exemplos práticos comentados para o melhor entendimento.

```
#Exemplo 1: Adicionando textos de fundo na janela
#abre uma janela vazia
tt <- tktoplevel()
#cria textos de fundo
tkpack(l1<-tklabel(tt,text="Texto 1"), l2<-tklabel(tt,text="Texto 2"))
#configura o alinhamento dos textos
tkpack.configure(l1, side="left")

#Exemplo 2: Utilizando widgets de comunicação
#abre uma janela vazia
tt <- tktoplevel()
#cria etiquetas de texto
label.widget <- tklabel(tt, text="Olá Mundo")
#cria um botão com ação
button.widget <- tkbutton(tt, text="Push",command=function()cat("OW!\n"))
#atualiza as widgets criadas
tkpack(label.widget, button.widget)
#destrói a aplicação
tkdestroy(tt)
```

Para maiores informações sobre aplicações do tcltk utilizando o R utilize o comando help(tcltk) no próprio R e leia os manuais das widgets existentes.

3 APLICAÇÃO

Com embasamento no capítulo 2, foi desenvolvido um pacote para utilização no R-2.4.0 utilizando recursos de criação de pacotes, tcltk e funções densidade de probabilidade. O pacote criado foi denominado **plott** e sua principal característica é a interação entre o gráfico de uma f.d.p.[11] e um painel onde são controlados os parâmetros desta distribuição.

3.1 O pacote PLOTT

O pacote plott foi desenvolvido utilizando todos os recursos comentados na metodologia deste trabalho, porém, também são utilizados outros dois pacotes que fazem integração e exibição de widgets, o rpanel e o tkrplot. O "rpanel" tem por objetivo fazer a comunicação entre as widgets e o "tkrplot" faz a divisão do aplicativo em duas áreas conforme figura 23.

Para instalar o pacote, basta acessar o cd-rom do anexo 7.2 ou no site *http://www.temich.com.br/temp* e fazer download do pacote para seu computador. Após instalado carregue e verifique o manual do pacote digitando os comandos:

```
library(plott)
help(plott)
```

O pacote plott é dividido em dois grupos de usabilidade:

- **Funções gerais –** Neste grupo temos duas funções (tabli e plotti). As utilidades destas funções estão mais ligadas à exibição de recursos criados com tcltk do que à aplicação estatística.
- **Funções específicas –** Neste grupo temos seis funções (normal, tstudent, quiquadrado, ffisher, exponencial e gama). O uso destas funções tem aplicação educacional, pois podemos estudar o comportamento de uma f.d.p. modificando os parâmetros e verificando o comportamento gráfico exibido no painel.

[11] f.d.p. – Função densidade de probabilidade

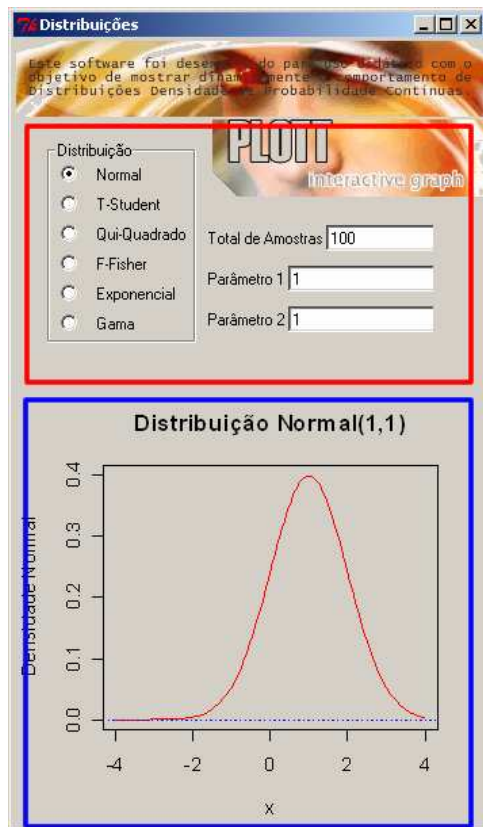
Todas as funções que geraram o pacote **plott** podem ser verificadas no anexo 7.3.

3.1.1 As funções gerais do pacote plott

O pacote plott possui duas funções gerais que são exibidas em um painel gráfico com duas áreas de dinamização (figura 23).

- **Área superior – Painel de controle/parâmetros (seleção vermelha):** Nesta área estão listadas as distribuições que fazem parte da função e ao selecionar uma distribuição no painel de controle, o painel de gráficos sincroniza automaticamente o formato desta distribuição. Também nesta área estão os campos de parâmetros das distribuições e o tamanho da amostra.

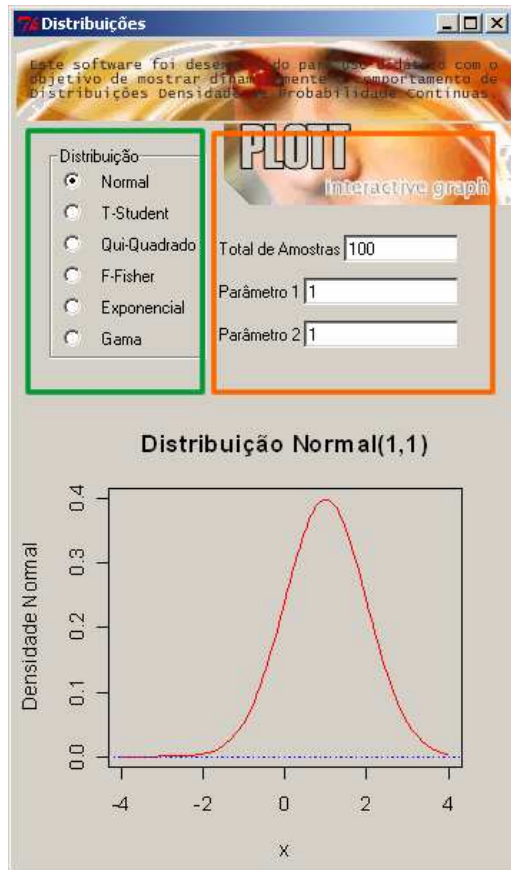
- **Área inferior – Painel de gráficos (seleção azul):** Este painel é o local de exibição gráfica, está em sincronia com o painel de controle/parâmetros.

Figura 23 – Áreas exibidas no painel gráfico

3.1.1.1 A função plotti

A função plotti é exibida com o uso do seguinte comando:

```
plotti()
```

A função plotti (figura 24) exibe dinamicamente o comportamento de seis f.d.p. (normal, t-Student, qui-quadrado, F, exponencial e gama) a partir da entrada dos valores dos parâmetros. A desvantagem ao uso deste painel é que a exibição dos parâmetros (seleção laranja) não é totalmente sincronizada com a escolha da distribuição (seleção verde), por isso foi citado anteriormente que as funções gerais deste pacote tem a finalidade de mostrar o que pode ser exibido com uso de recursos tcltk.

Figura 24 – Pacote plott (função plotti)

3.1.1.2 A função tabli

A função tabli é exibida com o uso do seguinte comando:

```
tabli()
```

Na figura 25 podemos observar a janela gráfica exibida pela função tabli, uma das funções gerais do pacote plott.

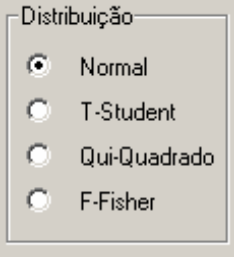
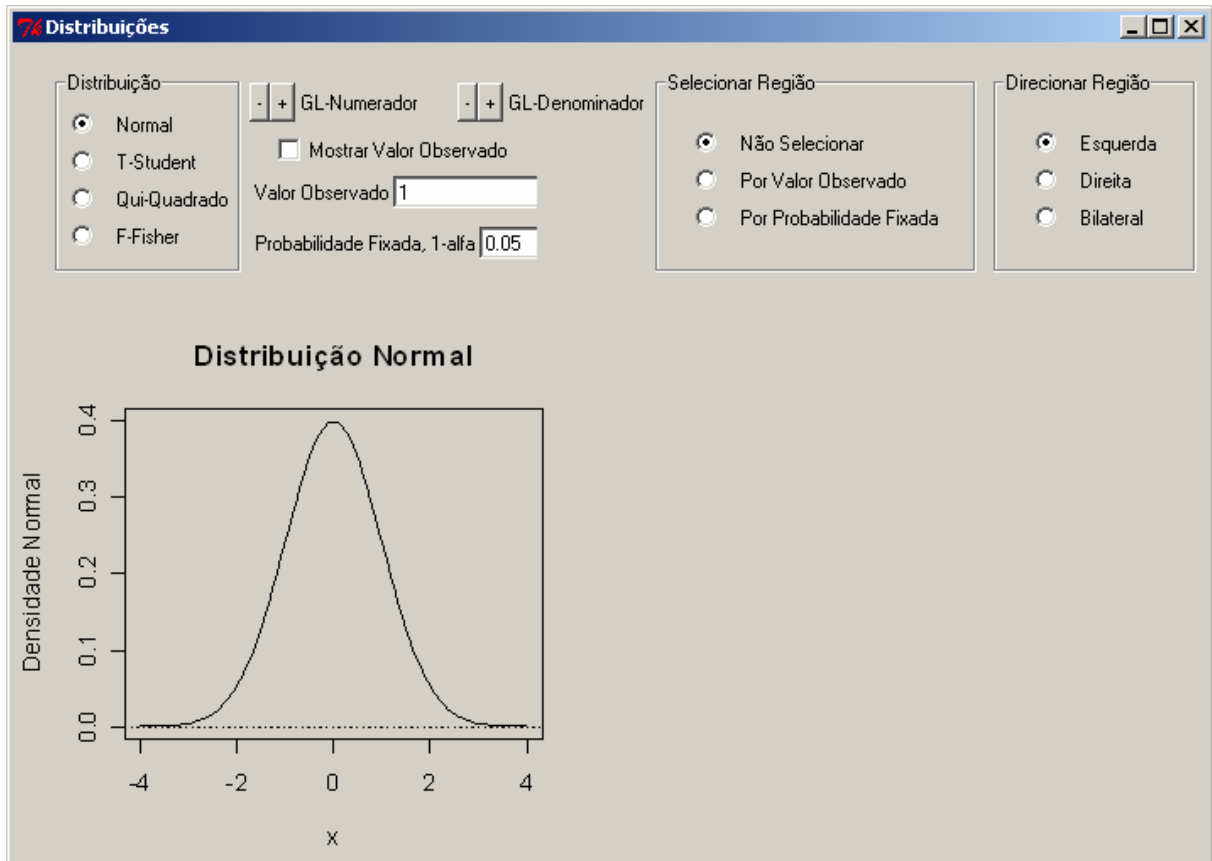
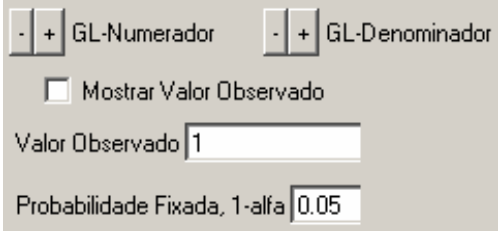
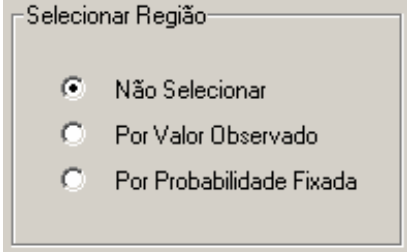
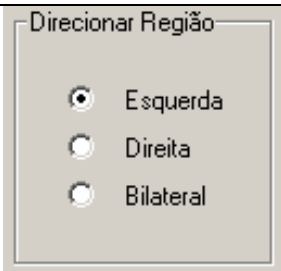



Figura 25 – Pacote plott (função tabli)

A função **tabli** é uma versão traduzida para português proveniente da função `rp.tables()` disponível no pacote **rpanel**. Esta função exibe um painel gráfico que simula uma tabela estatística para as distribuições normal, t-Student, qui-quadrado e F.

No painel de controle/parâmetros desta função temos várias opções de uso conforme a explicação na tabela 4.

Tabela 4 – Opções da função tabli()

Opção	**Explicação**
Distribuição: Normal, T-Student, Qui-Quadrado, F-Fisher	Nesta caixa de seleção podemos escolher uma distribuição para ser exibida. Podemos observar que será exibida somente uma distribuição por vez.

- + GL-Numerador - + GL-Denominador Mostrar Valor Observado Valor Observado 1 Probabilidade Fixada, 1-alfa 0.05	Aqui temos a opção de aumentar de diminuir os graus de liberdade da distribuição, exibir o valor observado pelo teste, entrar com o valor observado e especificar um nível de probabilidade.
Selecionar Região Não Selecionar Por Valor Observado Por Probabilidade Fixada	Nesta caixa de seleção podemos escolher a forma de exibição das áreas de rejeição do teste escolhido. Podemos exibir pelo valor observado, pela probabilidade fixada ou não exibir esta opção.
Direcionar Região Esquerda Direita Bilateral	Nesta caixa de seleção escolhemos a exibição da área de rejeição com opções de exibição à esquerda, direita ou ambos os lados.

3.1.2 As funções específicas do pacote plott

As funções específicas são casos especiais da função plotti. Como vimos na seção 3.1.1.1 a função plotti possui problemas na listagem dos parâmetros da distribuição. Para resolver este problema foram criadas separadamente funções que exibem o gráfico de cada f.d.p separadamente.

3.1.2.1 A distribuição Normal

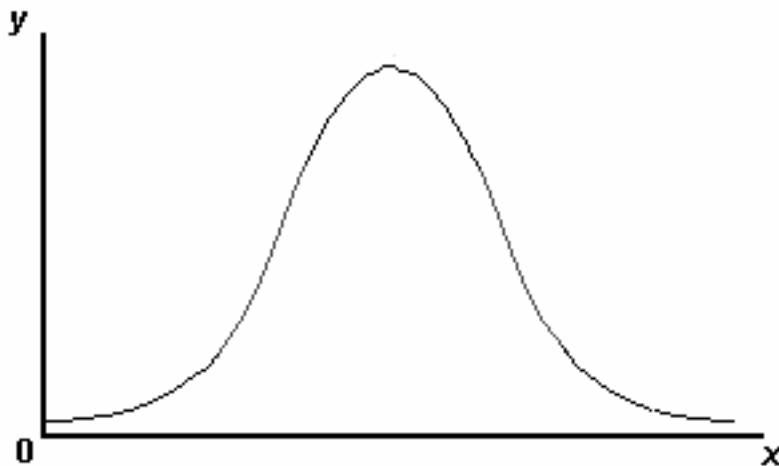
A **distribuição normal** é uma distribuição de probabilidade contínua que é simétrica e mesocúrtica (KASMIER, 1982). A curva que representa a distribuição normal de probabilidade tem o formato de um sino (figura 26) e é dada pela função 3.1.

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e\left[-\frac{(x-\mu)^2}{2\sigma^2}\right] \qquad (3.1)$$

Com a constante π = 3,1416 e os parâmetros:

μ = Média da distribuição;

σ = Desvio padrão da distribuição.

Figura 26 – Gráfico da distribuição normal

A distribuição normal é importante na inferência estatística por três razões distintas:

1 – As medidas produzidas em diversos processos aleatórios seguem esta distribuição (KASMIER, 1982).

2 – Probabilidades normais podem ser usadas frequentemente como aproximações de outras distribuições, tais como a *binomial* e *Poisson* (KASMIER, 1982).

3 – As distribuições de estatística da amostra tais como a média e a proporção frequentemente seguem a distribuição normal independente da distribuição da população (KASMIER, 1982).

No pacote plott temos o gráfico da distribuição **normal** que é exibida com o uso do seguinte comando:

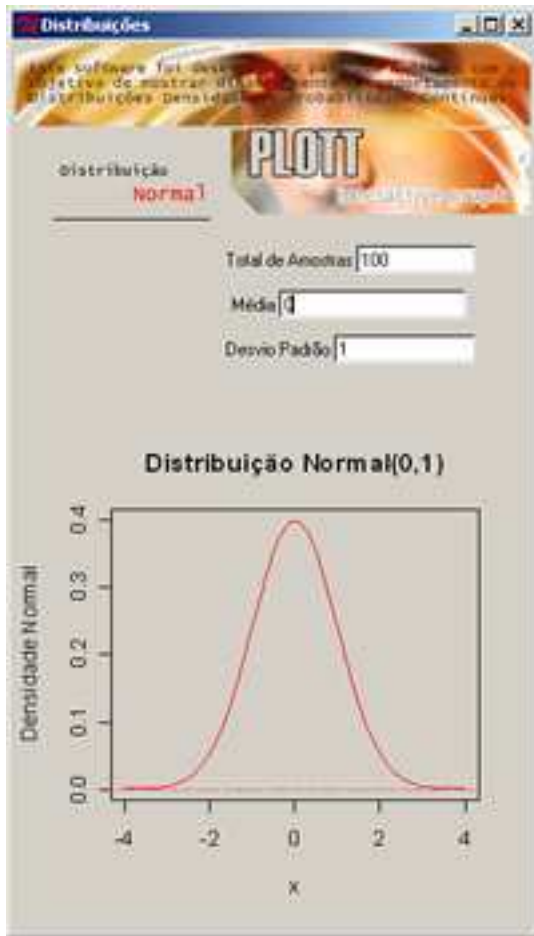
```
normal()
```

Ao exibir o painel da distribuição normal (figura 27) podemos notar que aparecem três campos para inserção de informações.

Total de amostras: Neste campo deve ser inserido a quantidade de amostras que devem seguir o modelo normal de probabilidade. Podemos notar que a curva desta distribuição perde sua suavidade a partir do momento que usamos um valor menor que 30, mostrando graficamente os resultados oferecidos pelo *teorema central do limite*.

Média: Neste campo deve ser inserido o valor da média da distribuição. Ao aumentar ou diminuir o valor da média podemos observar que o gráfico muda seu ponto central em relação à x. Quando aumentamos o valor de x notamos que o ponto central do gráfico muda para a direita e quando diminuímos a média o ponto central vai para esquerda.

Desvio padrão: Neste campo deve ser inserido o valor do desvio padrão da distribuição. Ao aumentar ou diminuir o valor do desvio padrão podemos notar uma mudança de comportamento em relação à altura afunilando ou alargando a curva exibida no gráfico.

Figura 27 – Pacote plott (distribuição normal)

Nota: Se utilizarmos os valores de média=0 e desvio padrão=1, estaremos exibindo a *distribuição normal padrão*.

3.1.2.2 A distribuição t-Student

Aparentemente a **distribuição t** é muito parecida com a distribuição normal. Ambas as distribuições têm curvas em formato de sino e são simétricas. Entretanto, a distribuição t tem maior área nas caudas e menor área no centro do que a distribuição normal (LEVINE et al., 1998). A distribuição t-Student é dada pela função 3.2.

$$f(x) = \frac{\Gamma[(\upsilon+1)/2]}{\Gamma(\upsilon/2)\sqrt{\pi\upsilon}}\left(1+\frac{x^2}{\upsilon}\right)^{-(\upsilon+1)/2} \qquad (3.2)$$

Com a constante π = 3,1416 e o parâmetro:

υ = Graus de liberdade.

Nota: A variância de uma amostra é dada por 3.3.

$$S^2 = \sum_{i=1}^{n}(X_i - \overline{X})^2 \qquad (3.3)$$

Sendo:

X_i = Valor observado;

$\overline{X}$ = Média amostral.

Portanto, para calcular S², primeiramente precisamos conhecer $\overline{X}$. Por esse motivo, podemos dizer que somente n-1 dos valores de amostras podem variar livremente. Isto é, existem n-1 graus de liberdade (LEVINE et al., 1998).

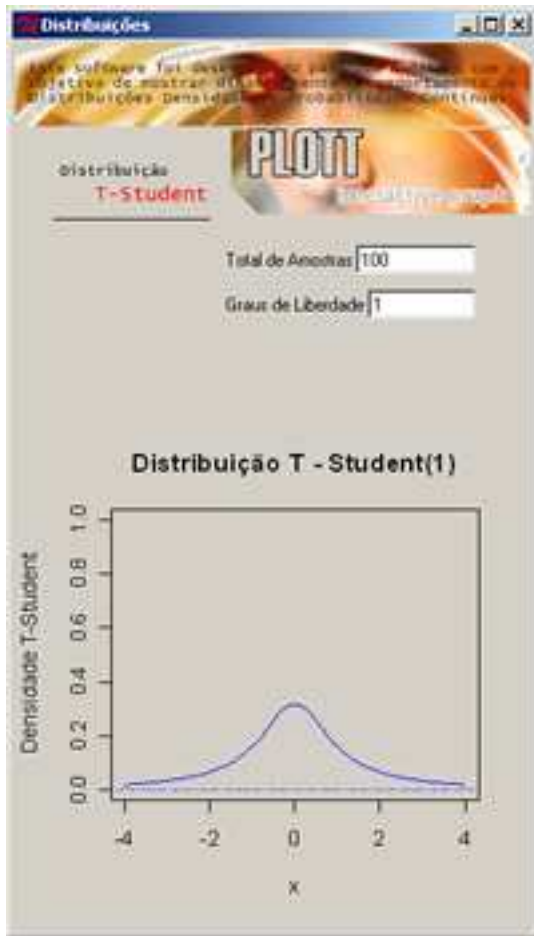
Para exemplificar o comportamento da distribuição t podemos utilizar no pacote plott o seguinte comando:

```
tstudent()
```

Ao abrir a janela gráfica (figura 28) podemos observar dois campos de inserção na área de controle/parâmetros.

Total de amostras: Neste campo deve ser inserido a quantidade de amostras que deve seguir a distribuição t. Como vimos no caso da distribuição normal aqui também iremos notar que a curva perde sua suavidade quando temos valores menores que 30.

Graus de liberdade: Neste campo deve ser inserido o número de graus de liberdade da distribuição. Observe que os graus de liberdade devem ser $0<\upsilon<\infty$. Outra questão importante que podemos observar é que conforme aumentam os graus de liberdade a distribuição t aproxima-se da distribuição normal.

Figura 28 – Pacote plott (distribuição t-Student)

3.1.2.3 A distribuição *X*

A distribuição **qui-quadrado** é denotada pelo simbolo ***X***. Esta distribuição é um caso particular, muito importante, da distribuição gama e é obtido no caso em que α=1/2 e *r*=υ/2 , onde υ é número de graus de liberdade da distribuição *X* (MEYER, 1978). Para entender melhor sobre α e *r*, leia o item 3.1.2.6.

A distribuição *X* é dada pela função 3.4.

$$f(x) = \frac{x^{(\upsilon/2)-1} e^{-x/2}}{2^{\upsilon/2}\Gamma(\upsilon/2)} \qquad (3.4)$$

Onde o parâmetro:

υ = Graus de liberdade.

Um caso bastante comum do uso da distribuição X é aquela em que desejamos testar se uma variável segue determinado modelo, mas desconhecemos um ou mais parâmetros da distribuição. Sendo assim, vamos utilizar a amostra para chegarmos às estimativas dos parâmetros desconhecidos, isto é, utilizando as próprias observações que dispomos, vamos obter estimativas que serão consideradas como valores dos parâmetros desconhecidos. Nesse tipo de caso, o número de graus de liberdade se altera para υ-1-*e*, com *e* representando o número de parâmetros que foram estimados (MAGALHÃES & LIMA, 2005).

A distribuição *X* não tem um formato gráfico fixo como a normal e a t-Student. Dependendo do número de graus de liberdade podemos terr uma curva convexa ou até mesmo uma curva côncava assimétrica. Para mostrar o comportamento da distribuição *X* podemos utilizar no pacote plott o seguinte comando:

```
quiquadrado()
```

Ao abrir a janela gráfica (figura 29) podemos observar dois campos de inserção na área de controle/parâmetros.

Total de amostras: Neste campo deve ser inserido a quantidade de amostras que devem seguir a distribuição *X*.

Graus de liberdade: Neste campo deve ser inserido o número de graus de liberdade da distribuição. Observe que os graus de liberdade devem ser $0<\upsilon<\infty$. Podemos ver também que conforme aumentam os graus de liberdade a distribuição *X* vai perdendo a forma convexa e tornando-se uma curva côncava assimétrica similar à distribuição normal.

Figura 29 – Pacote plott (distribuição *X*)

3.1.2.4 A distribuição F

Esta distribuição em conjunto com a qui-quadrado e t-Student, forma um conjunto de distribuições teóricas indispensáveis na resolução de problemas de inferência estatística. A **distribuição F** encontra um largo campo de aplicação em problemas relativos à análise de variância. É utilizada em larga escala na comparação de variâncias entre duas amostras tomadas independentemente da mesma população e normalmente distribuídas (KASMIER, 1982).

A distribuição F é dada pela função 3.5.

$$f(x) = \frac{\Gamma\left(\frac{\upsilon_1 + \upsilon_2}{2}\right) \upsilon_1^{\upsilon_1/2} \upsilon_2^{\upsilon_2/2} x^{(\upsilon_1/2)-1}}{\Gamma\left(\frac{\upsilon_1}{2}\right)\Gamma\left(\frac{\upsilon_2}{2}\right)(\upsilon_2 + \upsilon_1 x)^{\frac{\upsilon_1+\upsilon_2}{2}}} \qquad (3.5)$$

Onde os parâmetros:

υ_1 = Graus de liberdade do numerador;

υ_2 = Graus de liberdade do denominador.

Como a distribuição *X*, a distribuição F não tem um formato fixo. O formato da curva depende dos graus de liberdade fornecidos para numerador e denominador. Para mostrar o comportamento da distribuição F podemos utilizar no pacote plott o seguinte comando:

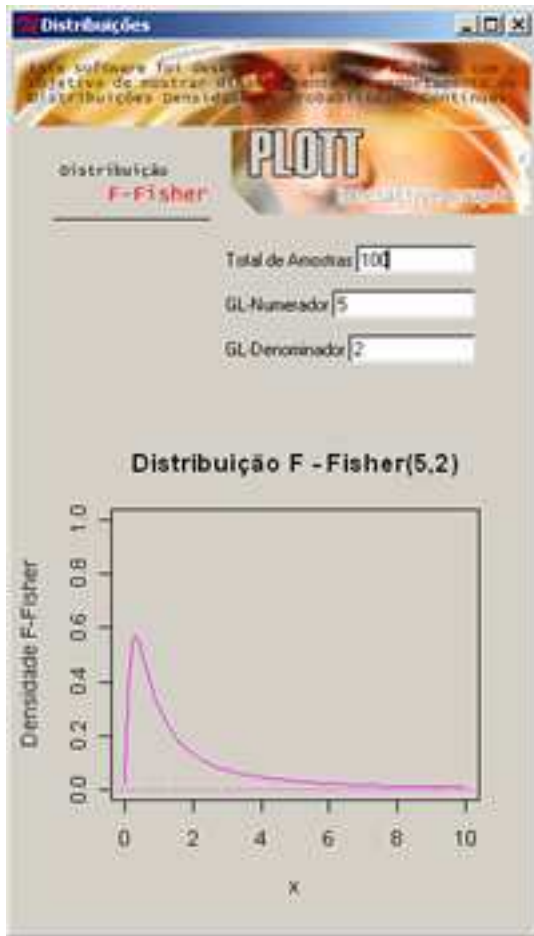
```
ffisher()
```

Ao abrir o painel da distribuição (figura 30) podemos observar três campos de inserção na área de controle/parâmetros.

Total de amostras: Neste campo deve ser inserido a quantidade de amostras que devem seguir a distribuição F. Conforme aumentamos o número de amostras a curva exibida torna-se mais suave.

Graus de liberdade do numerador: Neste campo deve ser inserido o número de graus de liberdade do numerador. Observe que os graus de liberdade devem ser $0<\upsilon<\infty$. Podemos ver também que conforme aumentam os graus de liberdade do numerador a curva exibida deixa o formato convexo e torna-se uma curva côncava assimétrica.

Graus de liberdade do denominador: Neste campo deve ser inserido o número de graus de liberdade do denominador. Observe que os graus de liberdade devem ser $0<\upsilon<\infty$. Podemos ver também que conforme aumentam os graus de liberdade do denominador a curva exibida vai afunilando-se e seu ponto máximo é elevado.

Figura 30 – Pacote plott (distribuição F)

3.1.2.5 A distribuição Exponencial

A **distribuição exponencial** é amplamente utilizada em linhas de espera ou teoria de filas para medir o tempo decorrido entre as chegadas, no âmbito da prestação de serviços como pedágios de pontes, nos caixas automáticos ou numa sala de emergência de um hospital. Esta distribuição é definida por um único parâmetro, λ , denominado taxa que é o tempo médio de chegadas por unidades de tempo (LEVINE et al., 1998).

A distribuição exponencial é dada pela função 3.6.

$$f(x) = \lambda e^{\lambda x} \qquad (3.6)$$

Onde o parâmetro:

λ = Taxa.

Uma particularidade desta distribuição é a característica conhecida como *falta de memória*. Esta característica deve-se à propriedade matemática desta distribuição que permite a translação da origem no cálculo de probabilidades (MAGALHÃES; LIMA, 2005), conforme exemplo 3.7.

$$P(X \geq 7 \mid X \geq 5) = P(X \geq 2) \qquad (3.7)$$

A distribuição exponencial tem o formato de uma curva convexa ao lado esquerdo do eixo x. Para mostrar o comportamento da distribuição exponencial podemos utilizar no pacote plott o seguinte comando:

```
exponencial()
```

Ao abrir a janela gráfica (figura 31) podemos observar dois campos de inserção na área de controle/parâmetros.

Total de amostras: Neste campo deve ser inserido a quantidade de amostras que devem seguir a distribuição exponencial. Conforme aumentamos o número de amostras a curva exibida torna-se mais suave.

Taxa: Neste campo deve ser inserida a taxa da distribuição. A taxa é o número médio de chegadas dividido pelo tempo. Podemos notar graficamente que conforme aumentamos o valor da taxa, o *ponto de sela* da curva aproxima-se do ponto 0 em x e em valores muito altos a curva perde sua suavidade transformando-se em uma reta paralela ao eixo y.

Figura 31 – Pacote plott (distribuição exponencial)

3.1.2.6 A distribuição Gama

A **distribuição gama** é a raiz de uma série de distribuições conhecidas como família exponencial e sua aplicação pode ser vista em casos que deseja-se saber o tempo para realizar determinada tarefa (KASMIER, 1982).

Os dois parâmetros da distribuição são α e β conhecidos por parâmetro de forma e parâmetro de escala respectivamente. A quantidade Γ(α) é o valor da função matemática padrão conhecida como *função gama*, definida pela integral 3.8.

$$\Gamma(\alpha) = \int_{0}^{\infty} t^{\alpha-1} e^{-t} \, dt \qquad (3.8)$$

E a distribuição gama é dada pela função 3.9.

$$f(x) = \frac{\alpha}{\Gamma(r)} (\alpha x)^{r-1} e^{-\alpha x} \qquad (3.9)$$

Onde os parâmetros:

α = Forma;

β = Escala.

A distribuição gama pode apresentar uma grande variedade de formas, dependendo, portanto, do parâmetro de forma α. Para valores de α muito altos, a distribuição gama tende à normal. O parâmetro de escala β, tem a função de esticar ou encolher, levando a curva para a direita ou esquerda.

Para mostrar o comportamento da distribuição gama podemos utilizar no pacote plott o seguinte comando:

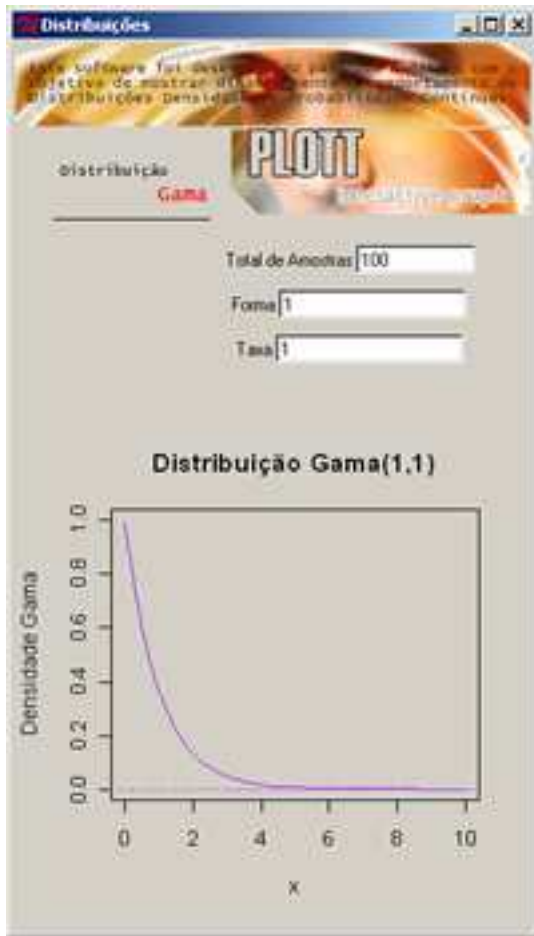
```
gama()
```

Ao abrir a janela gráfica (figura 32) podemos observar três campos de inserção na área de controle/parâmetros.

Total de amostras: Neste campo deve ser inserido a quantidade de amostras que devem seguir a distribuição gama. Conforme aumentamos o número de amostras a curva exibida torna-se mais suave.

Forma: Neste campo deve ser inserida a forma da distribuição. Ao aumentar o valor da forma percebemos que a distribuição tende a uma curva gaussiana.

Escala: Neste campo deve ser inserida a escala. Perceba que a curva estica à esquerda conforme aumentamos o valor da escala e encolhe a direita quando diminuímos o valor deste parâmetro.

Figura 32 – Pacote plott (distribuição gama)

4 CONSIDERAÇÕES FINAIS

A criação de pacotes com R-2.4.0 é facilitada pelas automações presentes na função package.skeleton porém um check-list deve ser feito para que não ocorram imprevistos. Como foi visto neste trabalho, algumas variáveis como plataforma computacional, sistema operacional e hardware podem interferir durante o processo de criação de pacotes.

O uso do tcltk torna as aplicações dinâmicas e agradáveis produzindo recursos que tornam o R-2.4.0 mais agradável ao usuário final. Apesar de ter 235 funções o tcltk pode ser aplicado com os recursos básicos desta linguagem/pacote, que são aproximadamente 15 funções.

Os resultados obtidos na produção do pacote plott mostram que o tcltk cria interação entre computador e usuário. Esta interação poderá ser ainda maior caso sejam desenvolvidas aplicações em tcltk utilizando outros recursos do R-2.4.0, tais como análise exploratória de dados, análise de experimentos, dados multivariados e outras.

Apesar de dinamizar todos os gráficos das f.d.p. (normal, t-Student, qui-quadrado, F, exponencial e gama) em uma única janela gráfica, a função plotti não mostrou bons resultados levando a avaliação de que as dificuldades de comunicação entre widgets pode aumentar conforme aumenta o número de painéis que se comunicam.

Além da oferta do pacote **plott** este trabalho pode ser utilizado como guia de consulta para utilizadores do R que queiram iniciar a construção de pacotes para ambientes Windows e Linux.

Por fim, sugiro como proposta futura a utilização de recursos mais complexos do tcltk que garantam uma maior interação entre usuário e computador em outras aplicações da estatística.

Assim, o presente trabalho procurou oferecer uma contribuição para usuários do software R.

6 REFERÊNCIAS BIBLIOGRÁFICAS

CAREY, V. **An introduction to the R packages mechanism.** Publicado em www.biostat.harvard.edu/courses/individual/bio271/lectures/L6/Rpkg.pdf, 2002.

DALGAARD, P. **A primer on the R-Tcl/tk package.** R News, 1(3):27-31, September 2001.

DALGAARD, P. **Changes to the R-Tcl/tk package.** R News, 1(3):25-27, December 2002.

FRANCA, A. **Tcltk Programação para Linux.** Rio de Janeiro: Editora Brasport, 2005.

FRASCATI, F. **Creare packages per R sotto Windows XP.** Publicado em http://cran.r-project.org/doc/contrib/Frascati-Rpackages.pdf, 2006.

KASMIER, L. J. **Estatística Aplicada à Economia e Administração.** São Paulo: Mcgraw-Hill do Brasil, 1982.

LEVINE, D. M.; BERENSON, M. L.; STEPHAN, D. **Estatística: Teoria e Aplicações.** Rio de Janeiro: LTC, 1998.

MAGALHÃES, M. N.; LIMA, A. C. P. **Noções de Probabilidade e Estatística.** São Paulo: Edusp, 2005.

MEYER, P. L. **Aplicações à Estatística.** Rio de Janeiro: LTC, 1978.

MONTEIRO, R. **Tcltk Guia de consulta rápida.** São Paulo: Editora Novatec, 2004.

MURDOCH, D. **Building R for Windows.** Publicado em http://www.murdoch-sutherland.com/Rtools/, 2005.

R Development Core Team. **An introduction to R.** Publicado em http://cran.r-project.org/doc/manuals/R-intro.html, 2006.

R Development Core Team. **Guidelines for Rd files.** Publicado em http://developer.r-project.org/Rds.html, 2005.

R Development Core Team. **Writing R Extensions.** Publicado em http://www.cran.r-project.org/doc/manuals/R-exts.pdf, 2006.

R Mailing lists. **The R-packages Archives.** Publicado em https://stat.ethz.ch/pipermail/r-packages/, 2005.

ROSSI, P. **Making R packages under Windows.** Publicado em http://http://gsbwww.uchicago.edu/fac/peter.rossi/research/bayes , 2005.

7 ANEXOS

7.1 Exemplo de arquivo .Rd

```
\name{soma}
\alias{soma}
\title{Soma de dois números}
\description{
  A função \code{soma} faz a soma de dois números.
}

\usage{
soma(a,b)
}

\arguments{
  \item{a}{primeiro número a ser somado}
  \item{b}{segundo número a ser somado}
}

\details{
  A função \code{soma} faz a soma de dois números, porém para bons
resultados
  é aconselhável seguir estas orientações.
}

\examples{
 soma(3,7)
 soma(25,25)
}

\keyword{ misc }
\eof
```

7.2 Cd-rom do pacote plott

7.3 Código fonte do pacote plott

```
plotti<-function (panel.plot = TRUE)
{
    require(tcltk)
    require(rpanel)
    require(tkrplot)
    tables.draw <- function(tables) {
        with(tables, {
            xobs <- as.numeric(xobs)
            prob <- as.numeric(prob)
            ngrid <- as.numeric(ngrid)
            degf1 <- as.numeric(degf1)
            degf2 <- as.numeric(degf2)
            if (distribution == "Normal") {
                xrange <- c(-4, 4)
                x <- seq(min(xrange[1], xobs * 1.1), max(xrange[2],
                  xobs * 1.1), length = ngrid)
                dens <- dnorm(x, degf1, degf2)
                ylim <- c(0, 0.4)
            }
            if (distribution == "T-Student") {
                xrange <- c(-4, 4)
                x <- seq(min(xrange[1], xobs * 1.1), max(xrange[2],
                  xobs * 1.1), length = ngrid)
                dens <- dt(x, degf1)
                ylim <- c(0, 0.4)
            }
            if (distribution == "Qui-Quadrado") {
                xrange <- c(0.01, degf1 + 3 * sqrt(2 * degf1))
                x <- seq(min(xrange[1], xobs * 1.1), max(xrange[2],
                  xobs * 1.1), length = ngrid)
                dens <- dchisq(x, degf1)
                ylim <- c(0, 0.4)
            }
            if (distribution == "F-Fisher") {
                xrange <- c(0.01, 10)
                x <- seq(min(xrange[1], xobs * 1.1), max(xrange[2],
                  xobs * 1.1), length = ngrid)
                dens <- df(x, degf1, degf2)
                ylim <- c(0, 1)
            }
            if (distribution == "Exponencial") {
                xrange <- c(0.01, 10)
                x <- seq(min(xrange[1], xobs * 1.1), max(xrange[2],
                  xobs * 1.1), length = ngrid)
                dens <- dexp(x, degf1)
                ylim <- c(0, 1)
            }
            if (distribution == "Gama") {
                xrange <- c(0.01, 10)
                x <- seq(min(xrange[1], xobs * 1.1), max(xrange[2],
                  xobs * 1.1), length = ngrid)
                dens <- dgamma(x, degf1, degf2)
                ylim <- c(0, 1)
            }
            plot(x, dens, type = "l", ylim = ylim, ylab =
paste("Densidade",
                distribution), col = "red")
            abline(h = 0, lty = 3, col = "blue")
            if (distribution == "Normal")
```

```
                title.text <- paste("Distribuição Normal(", degf1,
                  ",", degf2, ")", sep = "")
            if (distribution == "T-Student")
                title.text <- paste("Distribuição T - Student(",
                  degf1, ")", sep = "")
            if (distribution == "Qui-Quadrado")
                title.text <- paste("Distribuição Qui-Quadrado(",
                  degf1, ")", sep = "")
            if (distribution == "F-Fisher")
                title.text <- paste("Distribuição F - Fisher(",
                  degf1, ",", degf2, ")", sep = "")
            if (distribution == "Exponencial")
                title.text <- paste("Distribuição Exponencial(",
                  degf1, ")", sep = "")
            if (distribution == "Gama")
                title.text <- paste("Distribuição Gama(", degf1,
                  ",", degf2, ")", sep = "")
            title(title.text)
        })
        tables
    }
    tables.redraw <- function(object) {
        rp.tkrreplot(object, plot)
        object
    }
    panel.name <- rp.panelname()
    if (panel.plot && require(tkrplot)) {
        tables.panel <- rp.control("Distribuições", size = c(350,
            600), realname = panel.name, xobs = 1, prob = 0.05,
            distribution = "Normal", degf1 = 1, degf2 = 1, ngrid = 100)
        tables.panel <- rp.radiogroup(tables.panel, distribution,
            c("Normal", "T-Student", "Qui-Quadrado", "F-Fisher",
                "Exponencial", "Gama"), title = "Distribuição",
            action = tables.redraw, pos = c(25, 75, 110, 150))
        title.ngrid <- "Total"
        title.defg1 <- "Parâmetro 1"
        title.defg2 <- "Parâmetro 2"
        tables.panel <- rp.textentry(tables.panel, ngrid, tables.redraw,
            title = (paste(title.ngrid)), pos = c(140, 135, 170,
                25))
        tables.panel <- rp.textentry(tables.panel, degf1, tables.redraw,
            title = (paste(title.defg1)), pos = c(140, 165, 170,
                25))
        tables.panel <- rp.textentry(tables.panel, degf2, tables.redraw,
            title = (paste(title.defg2)), pos = c(140, 195, 170,
                25))
        click <- function(tables.panel, x, y) {
            print(c(x, y))
            tables.panel
        }
        image.file <- file.path(system.file(package = "plott"),
            "images", "topo1.gif")
        tables.panel <- rp.image(tables.panel, image.file, pos = c(0,
            0, 350, 60), id = "topo1", action = click)
        image.file <- file.path(system.file(package = "plott"),
            "images", "topo2.gif")
        tables.panel <- rp.image(tables.panel, image.file, pos = c(145,
            58, 205, 61), id = "topo2", action = click)
        tables.panel <- rp.tkrplot(tables.panel, plot, plotfun =
tables.draw,
            pos = c(0, 250, 700, 525))
```

```
    }
    else {
        if (panel.plot)
            rp.messagebox("O pacote TKRPLOT não está instalado! Este
aplicativo abrirá em 2 janelas separadas.")
        tables.panel <- rp.control("Distribuições", size = c(350,
            250), realname = panel.name, xobs = 1, prob = 0.05,
            distribution = "Normal", degf1 = 1, degf2 = 1, ngrid = 100)
        tables.panel <- rp.radiogroup(tables.panel, distribution,
            c("Normal", "T-Student", "Qui-Quadrado", "F-Fisher",
                "Exponencial", "Gama"), title = "Distribuição",
            action = tables.draw, pos = c(25, 75, 110, 150))
        title.ngrid <- "Total"
        title.defg1 <- "Parâmetro 1"
        title.defg2 <- "Parâmetro 2"
        tables.panel <- rp.textentry(tables.panel, ngrid, tables.draw,
            title = (paste(title.ngrid)), pos = c(140, 135, 170,
                25))
        tables.panel <- rp.textentry(tables.panel, degf1, tables.draw,
            title = (paste(title.defg1)), pos = c(140, 165, 170,
                25))
        tables.panel <- rp.textentry(tables.panel, degf2, tables.draw,
            title = (paste(title.defg2)), pos = c(140, 195, 170,
                25))
        click <- function(tables.panel, x, y) {
            print(c(x, y))
            tables.panel
        }
        image.file1 <- file.path(system.file(package = "plott"),
            "images", "topo1.gif")
        tables.panel <- rp.image(tables.panel, image.file1, pos = c(0,
            0, 350, 60), id = "topo1", action = click)
        image.file2 <- file.path(system.file(package = "plott"),
            "images", "topo2.gif")
        tables.panel <- rp.image(tables.panel, image.file2, pos = c(145,
            58, 205, 61), id = "topo2", action = click)
        tables.panel <- rp.tkrplot(tables.panel, plot, plotfun =
tables.draw,
            pos = c(0, 250, 700, 525))
        rp.do(tables.panel, tables.draw)
    }
}

tabli<- function (panel.plot = TRUE)
{
    require(tcltk)
    require(rpanel)
    require(tkrplot)
    tables.draw <- function(tables) {
        with(tables, {
            xobs <- as.numeric(xobs)
            prob <- as.numeric(prob)
            ngrid <- 100
            if (distribution == "Normal") {
                xrange <- c(-4, 4)
                x <- seq(min(xrange[1], xobs * 1.1), max(xrange[2],
                  xobs * 1.1), length = ngrid)
                dens <- dnorm(x)
                ylim <- c(0, 0.4)
                pval <- pnorm(xobs)
                pshade <- min(pval, 1 - pval)
```

```
            qts <- qnorm(c(prob, 1 - prob, prob/2, 1 - prob/2,
              pshade, 1 - pshade))
        }
        if (distribution == "T-Student") {
            xrange <- c(-4, 4)
            x <- seq(min(xrange[1], xobs * 1.1), max(xrange[2],
              xobs * 1.1), length = ngrid)
            dens <- dt(x, degf1)
            ylim <- c(0, 0.4)
            pval <- pt(xobs, degf1)
            pshade <- min(pval, 1 - pval)
            qts <- qt(c(prob, 1 - prob, prob/2, 1 - prob/2,
              pshade, 1 - pshade), degf1)
        }
        if (distribution == "Qui-Quadrado") {
            xrange <- c(0.01, degf1 + 3 * sqrt(2 * degf1))
            x <- seq(min(xrange[1], xobs * 1.1), max(xrange[2],
              xobs * 1.1), length = ngrid)
            dens <- dchisq(x, degf1)
            ylim <- c(0, 0.4)
            pval <- pchisq(xobs, degf1)
            pshade <- min(pval, 1 - pval)
            qts <- qchisq(c(prob, 1 - prob, prob/2, 1 - prob/2,
              pshade, 1 - pshade), degf1)
        }
        if (distribution == "F-Fisher") {
            xrange <- c(0.01, 10)
            x <- seq(min(xrange[1], xobs * 1.1), max(xrange[2],
              xobs * 1.1), length = ngrid)
            dens <- df(x, degf1, degf2)
            ylim <- c(0, 1)
            pval <- pf(xobs, degf1, degf2)
            pshade <- min(pval, 1 - pval)
            qts <- qf(c(prob, 1 - prob, prob/2, 1 - prob/2,
              pshade, 1 - pshade), degf1, degf2)
        }
        plot(x, dens, type = "l", ylim = ylim, ylab =
paste("Densidade",
            distribution))
        abline(h = 0, lty = 3)
        if (distribution == "Normal")
            title.text <- "Distribuição Normal"
        if (distribution == "T-Student")
            title.text <- paste("Distribuição T-Student(",
              degf1, ")", sep = "")
        if (distribution == "Qui-Quadrado")
            title.text <- paste("Distribuição Qui-Quadrado(",
              degf1, ")", sep = "")
        if (distribution == "F-Fisher")
            title.text <- paste("Distribuição F-Fisher(",
              degf1, ",", degf2, ")", sep = "")
        xobs <- as.numeric(xobs)
        if (observed.value.showing & !is.na(xobs)) {
            lines(rep(xobs, 2), c(0, ylim[2] * 0.95), lty = 2)
            text(xobs, ylim[2] * 0.97, signif(xobs, 4))
            title.text <- paste(title.text, ";  xobs =",
              round(xobs, 2))
        }
        if (tail.area != "Não Selecionar") {
            if (tail.area == "Por Probabilidade Fixada") {
              if (tail.direction == "Esquerda")
```

```
                shade <- c(qts[1], NA)
              else if (tail.direction == "Direita")
                shade <- c(NA, qts[2])
              else shade <- qts[3:4]
              title.text <- paste(title.text, ";  p =", round(prob,
                3))
            }
            else {
              if (tail.direction == "Esquerda")
                shade <- c(xobs, NA)
              else if (tail.direction == "Direita") {
                shade <- c(NA, xobs)
                pval <- 1 - pval
              }
              else {
                shade <- qts[5:6]
                pval <- 2 * min(pval, 1 - pval)
              }
              title.text <- paste(title.text, ";  pval =",
                round(pval, 3))
            }
            if (!is.na(shade[1])) {
              ind <- max((1:ngrid)[x < shade[1]])
              intp <- (shade[1] - x[ind])/(x[ind + 1] - x[ind])
              dend <- (1 - intp) * dens[ind] + intp * dens[ind +
                1]
              dt <- c(dens[x < shade[1]], dend)
              xt <- c(x[x < shade[1]], shade[1])
              polygon(c(xt, rev(xt)), c(dt, rep(0, length(xt))),
                density = -1, col = "blue", border = "blue")
            }
            if (!is.na(shade[2])) {
              ind <- min((1:ngrid)[x > shade[2]])
              intp <- (shade[2] - x[ind - 1])/(x[ind] - x[ind -
                1])
              dend <- (1 - intp) * dens[ind - 1] + intp *
                dens[ind]
              dt <- c(dend, dens[x > shade[2]])
              xt <- c(shade[2], x[x > shade[2]])
              polygon(c(xt, rev(xt)), c(dt, rep(0, length(xt))),
                density = -1, col = "red", border = "red")
            }
        }
        title(title.text)
    })
    tables
}
tables.redraw <- function(object) {
    rp.tkrreplot(object, plot)
    object
}
panel.name <- rp.panelname()
if (panel.plot && require(tkrplot)) {
    tables.panel <- rp.control("Distribuições", size = c(720,
        500), realname = panel.name, xobs = 1, prob = 0.05,
        distribution = "Normal", degf1 = 5, degf2 = 30)
    tables.panel <- rp.radiogroup(tables.panel, distribution,
        c("Normal", "T-Student", "Qui-Quadrado", "F-Fisher"),
        title = "Distribuição", action = tables.redraw, pos = c(25,
            15, 110, 120))
    tables.panel <- rp.doublebutton(tables.panel, degf1,
```

```
            1, range = c(1, NA), title = "GL-Numerador", action =
tables.redraw,
            pos = c(140, 10, 100, 50))
        tables.panel <- rp.doublebutton(tables.panel, degf2,
            1, range = c(1, NA), title = "GL-Denominador", action =
tables.redraw,
            pos = c(260, 10, 120, 50))
        tables.panel <- rp.checkbox(tables.panel, observed.value.showing,
            title = "Mostrar Valor Observado", action = tables.redraw,
            pos = c(140, 50, 170, 25))
        tables.panel <- rp.textentry(tables.panel, xobs, tables.redraw,
            "Valor Observado", pos = c(140, 75, 170, 25))
        tables.panel <- rp.textentry(tables.panel, prob, tables.redraw,
            "Probabilidade Fixada, 1-alfa", pos = c(140, 105,
                170, 25))
        tables.panel <- rp.radiogroup(tables.panel, tail.area,
            c("Não Selecionar", "Por Valor Observado", "Por Probabilidade
Fixada"),
            title = "Selecionar Região", action = tables.redraw,
            pos = c(380, 15, 190, 120))
        tables.panel <- rp.radiogroup(tables.panel, tail.direction,
            c("Esquerda", "Direita", "Bilateral"), title = "Direcionar
Região",
            action = tables.redraw, pos = c(580, 15, 120, 120))
        tables.panel <- rp.tkrplot(tables.panel, plot, plotfun =
tables.draw,
            pos = c(0, 150, 625, 525))
    }
    else {
        if (panel.plot)
            rp.messagebox("O pacote TKRPLOT não está instalado! Este
aplicativo abrirá em 2 janelas separadas.")
        tables.panel <- rp.control("Distribuições", size = c(720,
            150), realname = panel.name, xobs = 1, prob = 0.05,
            distribution = "Normal", degf1 = 5, degf2 = 30)
        tables.panel <- rp.radiogroup(tables.panel, distribution,
            c("Normal", "T-Student", "Qui-Quadrado", "F-Fisher"),
            title = "Distribuição", action = tables.draw, pos = c(25,
                15, 110, 120))
        tables.panel <- rp.doublebutton(tables.panel, degf1,
            1, range = c(1, NA), title = "GL-Numerador", action =
tables.draw,
            pos = c(140, 10, 100, 50))
        tables.panel <- rp.doublebutton(tables.panel, degf2,
            1, range = c(1, NA), title = "GL-Denominador", action =
tables.draw,
            pos = c(260, 10, 120, 50))
        tables.panel <- rp.checkbox(tables.panel, observed.value.showing,
            title = "Mostrar Valor Observado", action = tables.draw,
            pos = c(140, 50, 170, 25))
        tables.panel <- rp.textentry(tables.panel, xobs, tables.draw,
            "Valor Observado", pos = c(140, 75, 170, 25))
        tables.panel <- rp.textentry(tables.panel, prob, tables.draw,
            "Probabilidade Fixada, 1-alfa", pos = c(140, 105,
                170, 25))
        tables.panel <- rp.radiogroup(tables.panel, tail.area,
            c("Não Selecionar", "Por Valor Observado", "Por Probabilidade
Fixada"),
            title = "Selecionar Região", action = tables.draw,
            pos = c(380, 15, 190, 120))
        tables.panel <- rp.radiogroup(tables.panel, tail.direction,
```

```
            c("Esquerda", "Direita", "Bilateral"), title = "Direcionar
Região",
            action = tables.draw, pos = c(580, 15, 120, 120))
        rp.do(tables.panel, tables.draw)
    }
}

normal<- function (panel.plot = TRUE)
{
    require(tcltk)
    require(rpanel)
    require(tkrplot)
    tables.draw <- function(tables) {
        with(tables, {
            xobs <- as.numeric(xobs)
            prob <- as.numeric(prob)
            ngrid <- as.numeric(ngrid)
            degf1 <- as.numeric(degf1)
            degf2 <- as.numeric(degf2)
            if (distribution == "Normal") {
                xrange <- c(-4, 4)
                x <- seq(min(xrange[1], xobs * 1.1), max(xrange[2],
                  xobs * 1.1), length = ngrid)
                dens <- dnorm(x, degf1, degf2)
                ylim <- c(0, 0.4)
            }
            plot(x, dens, type = "l", ylim = ylim, ylab =
paste("Densidade",
                distribution), col = "red")
            abline(h = 0, lty = 3, col = "red")
            if (distribution == "Normal")
                title.text <- paste("Distribuição Normal(", degf1,
                  ",", degf2, ")", sep = "")
            title(title.text)
        })
        tables
    }
    tables.redraw <- function(object) {
        rp.tkrreplot(object, plot)
        object
    }
    panel.name <- rp.panelname()
    if (panel.plot && require(tkrplot)) {
        tables.panel <- rp.control("Distribuições", size = c(350,
            600), realname = panel.name, xobs = 1, prob = 0.05,
            distribution = "Normal", degf1 = 1, degf2 = 1, ngrid = 100)
        title.ngrid <- "Total de Amostras"
        title.defg1 <- "Média"
        title.defg2 <- "Desvio Padrão"
        tables.panel <- rp.textentry(tables.panel, ngrid, tables.redraw,
            title = (paste(title.ngrid)), pos = c(140, 135, 170,
                25))
        tables.panel <- rp.textentry(tables.panel, degf1, tables.redraw,
            title = (paste(title.defg1)), pos = c(140, 165, 170,
                25))
        tables.panel <- rp.textentry(tables.panel, degf2, tables.redraw,
            title = (paste(title.defg2)), pos = c(140, 195, 170,
                25))
        click <- function(tables.panel, x, y) {
            print(c(x, y))
            tables.panel
```

```
        }
        image.file <- file.path(system.file(package = "plott"),
            "images", "normal.gif")
        tables.panel <- rp.image(tables.panel, image.file, pos = c(25,
            75, 110, 150), id = "topo0", action = click)
        image.file <- file.path(system.file(package = "plott"),
            "images", "topo1.gif")
        tables.panel <- rp.image(tables.panel, image.file, pos = c(0,
            0, 350, 60), id = "topo1", action = click)
        image.file <- file.path(system.file(package = "plott"),
            "images", "topo2.gif")
        tables.panel <- rp.image(tables.panel, image.file, pos = c(145,
            58, 205, 61), id = "topo2", action = click)
        tables.panel <- rp.tkrplot(tables.panel, plot, plotfun =
tables.draw,
            pos = c(0, 250, 700, 525))
    }
    else {
        if (panel.plot)
            rp.messagebox("O pacote TKRPLOT não está instalado! Este
aplicativo abrirá em 2 janelas separadas.")
        tables.panel <- rp.control("Distribuições", size = c(350,
            250), realname = panel.name, xobs = 1, prob = 0.05,
            distribution = "Normal", degf1 = 1, degf2 = 1, ngrid = 100)
        title.ngrid <- "Total de Amostras"
        title.defg1 <- "Média"
        title.defg2 <- "Desvio Padrão"
        tables.panel <- rp.textentry(tables.panel, ngrid, tables.draw,
            title = (paste(title.ngrid)), pos = c(140, 135, 170,
                25))
        tables.panel <- rp.textentry(tables.panel, degf1, tables.draw,
            title = (paste(title.defg1)), pos = c(140, 165, 170,
                25))
        tables.panel <- rp.textentry(tables.panel, degf2, tables.draw,
            title = (paste(title.defg2)), pos = c(140, 195, 170,
                25))
        click <- function(tables.panel, x, y) {
            print(c(x, y))
            tables.panel
        }
        image.file <- file.path(system.file(package = "plott"),
            "images", "normal.gif")
        tables.panel <- rp.image(tables.panel, image.file, pos = c(25,
            75, 110, 150), id = "topo0", action = click)
        image.file1 <- file.path(system.file(package = "plott"),
            "images", "topo1.gif")
        tables.panel <- rp.image(tables.panel, image.file1, pos = c(0,
            0, 350, 60), id = "topo1", action = click)
        image.file2 <- file.path(system.file(package = "plott"),
            "images", "topo2.gif")
        tables.panel <- rp.image(tables.panel, image.file2, pos = c(145,
            58, 205, 61), id = "topo2", action = click)
        tables.panel <- rp.tkrplot(tables.panel, plot, plotfun =
tables.draw,
            pos = c(0, 250, 700, 525))
        rp.do(tables.panel, tables.draw)
    }
}

tstudent<- function (panel.plot = TRUE)
{
```

```
    require(tcltk)
    require(rpanel)
    require(tkrplot)
    tables.draw <- function(tables) {
        with(tables, {
            xobs <- as.numeric(xobs)
            prob <- as.numeric(prob)
            ngrid <- as.numeric(ngrid)
            degf1 <- as.numeric(degf1)
            degf2 <- as.numeric(degf2)
            if (distribution == "T-Student") {
                xrange <- c(-4, 4)
                x <- seq(min(xrange[1], xobs * 1.1), max(xrange[2],
                  xobs * 1.1), length = ngrid)
                dens <- dt(x, degf1)
                ylim <- c(0, 1)
            }
            plot(x, dens, type = "l", ylim = ylim, ylab =
paste("Densidade",
                distribution), col = "blue")
            abline(h = 0, lty = 3, col = "blue")
            if (distribution == "T-Student")
                title.text <- paste("Distribuição T - Student(",
                  degf1, ")", sep = "")
            title(title.text)
        })
        tables
    }
    tables.redraw <- function(object) {
        rp.tkrreplot(object, plot)
        object
    }
    panel.name <- rp.panelname()
    if (panel.plot && require(tkrplot)) {
        tables.panel <- rp.control("Distribuições", size = c(350,
            600), realname = panel.name, xobs = 1, prob = 0.05,
            distribution = "T-Student", degf1 = 1, degf2 = 1,
            ngrid = 100)
        title.ngrid <- "Total de Amostras"
        title.defg1 <- "Graus de Liberdade"
        tables.panel <- rp.textentry(tables.panel, ngrid, tables.redraw,
            title = (paste(title.ngrid)), pos = c(140, 135, 170,
                25))
        tables.panel <- rp.textentry(tables.panel, degf1, tables.redraw,
            title = (paste(title.defg1)), pos = c(140, 165, 170,
                25))
        click <- function(tables.panel, x, y) {
            print(c(x, y))
            tables.panel
        }
        image.file <- file.path(system.file(package = "plott"),
            "images", "student.gif")
        tables.panel <- rp.image(tables.panel, image.file, pos = c(25,
            75, 110, 150), id = "topo0", action = click)
        image.file <- file.path(system.file(package = "plott"),
            "images", "topo1.gif")
        tables.panel <- rp.image(tables.panel, image.file, pos = c(0,
            0, 350, 60), id = "topo1", action = click)
        image.file <- file.path(system.file(package = "plott"),
            "images", "topo2.gif")
        tables.panel <- rp.image(tables.panel, image.file, pos = c(145,
```

```
            58, 205, 61), id = "topo2", action = click)
        tables.panel <- rp.tkrplot(tables.panel, plot, plotfun =
tables.draw,
            pos = c(0, 250, 700, 525))
    }
    else {
        if (panel.plot)
            rp.messagebox("O pacote TKRPLOT não está instalado! Este
aplicativo abrirá em 2 janelas separadas.")
        tables.panel <- rp.control("Distribuições", size = c(350,
            250), realname = panel.name, xobs = 1, prob = 0.05,
            distribution = "T-Student", degf1 = 1, degf2 = 1,
            ngrid = 100)
        title.ngrid <- "Total de Amostras"
        title.defg1 <- "Graus de Liberdade"
        tables.panel <- rp.textentry(tables.panel, ngrid, tables.draw,
            title = (paste(title.ngrid)), pos = c(140, 135, 170,
                25))
        tables.panel <- rp.textentry(tables.panel, degf1, tables.draw,
            title = (paste(title.defg1)), pos = c(140, 165, 170,
                25))
        click <- function(tables.panel, x, y) {
            print(c(x, y))
            tables.panel
        }
        image.file <- file.path(system.file(package = "plott"),
            "images", "student.gif")
        tables.panel <- rp.image(tables.panel, image.file, pos = c(25,
            75, 110, 150), id = "topo0", action = click)
        image.file1 <- file.path(system.file(package = "plott"),
            "images", "topo1.gif")
        tables.panel <- rp.image(tables.panel, image.file1, pos = c(0,
            0, 350, 60), id = "topo1", action = click)
        image.file2 <- file.path(system.file(package = "plott"),
            "images", "topo2.gif")
        tables.panel <- rp.image(tables.panel, image.file2, pos = c(145,
            58, 205, 61), id = "topo2", action = click)
        tables.panel <- rp.tkrplot(tables.panel, plot, plotfun =
tables.draw,
            pos = c(0, 250, 700, 525))
        rp.do(tables.panel, tables.draw)
    }
}

quiquadrado<- function (panel.plot = TRUE)
{
    require(tcltk)
    require(rpanel)
    require(tkrplot)
    tables.draw <- function(tables) {
        with(tables, {
            xobs <- as.numeric(xobs)
            prob <- as.numeric(prob)
            ngrid <- as.numeric(ngrid)
            degf1 <- as.numeric(degf1)
            degf2 <- as.numeric(degf2)
            if (distribution == "Qui-Quadrado") {
                xrange <- c(0.01, degf1 + 3 * sqrt(2 * degf1))
                x <- seq(min(xrange[1], xobs * 1.1), max(xrange[2],
                  xobs * 1.1), length = ngrid)
                dens <- dchisq(x, degf1)
```

```
                ylim <- c(0, 1)
            }
            plot(x, dens, type = "l", ylim = ylim, ylab =
paste("Densidade",
                distribution), col = "black")
            abline(h = 0, lty = 3, col = "black")
            if (distribution == "Qui-Quadrado")
                title.text <- paste("Distribuição Qui-Quadrado(",
                  degf1, ")", sep = "")
            title(title.text)
        })
        tables
    }
    tables.redraw <- function(object) {
        rp.tkrreplot(object, plot)
        object
    }
    panel.name <- rp.panelname()
    if (panel.plot && require(tkrplot)) {
        tables.panel <- rp.control("Distribuições", size = c(350,
            600), realname = panel.name, xobs = 1, prob = 0.05,
            distribution = "Qui-Quadrado", degf1 = 1, degf2 = 1,
            ngrid = 100)
        title.ngrid <- "Total de Amostras"
        title.defg1 <- "Graus de Liberdade"
        tables.panel <- rp.textentry(tables.panel, ngrid, tables.redraw,
            title = (paste(title.ngrid)), pos = c(140, 135, 170,
                25))
        tables.panel <- rp.textentry(tables.panel, degf1, tables.redraw,
            title = (paste(title.defg1)), pos = c(140, 165, 170,
                25))
        click <- function(tables.panel, x, y) {
            print(c(x, y))
            tables.panel
        }
        image.file <- file.path(system.file(package = "plott"),
            "images", "quadrado.gif")
        tables.panel <- rp.image(tables.panel, image.file, pos = c(25,
            75, 110, 150), id = "topo0", action = click)
        image.file <- file.path(system.file(package = "plott"),
            "images", "topo1.gif")
        tables.panel <- rp.image(tables.panel, image.file, pos = c(0,
            0, 350, 60), id = "topo1", action = click)
        image.file <- file.path(system.file(package = "plott"),
            "images", "topo2.gif")
        tables.panel <- rp.image(tables.panel, image.file, pos = c(145,
            58, 205, 61), id = "topo2", action = click)
        tables.panel <- rp.tkrplot(tables.panel, plot, plotfun =
tables.draw,
            pos = c(0, 250, 700, 525))
    }
    else {
        if (panel.plot)
            rp.messagebox("O pacote TKRPLOT não está instalado! Este
aplicativo abrirá em 2 janelas separadas.")
        tables.panel <- rp.control("Distribuições", size = c(350,
            250), realname = panel.name, xobs = 1, prob = 0.05,
            distribution = "Qui-Quadrado", degf1 = 1, degf2 = 1,
            ngrid = 100)
        title.ngrid <- "Total de Amostras"
        title.defg1 <- "Graus de Liberdade"
```

```
        tables.panel <- rp.textentry(tables.panel, ngrid, tables.draw,
            title = (paste(title.ngrid)), pos = c(140, 135, 170,
                25))
        tables.panel <- rp.textentry(tables.panel, degf1, tables.draw,
            title = (paste(title.defg1)), pos = c(140, 165, 170,
                25))
        click <- function(tables.panel, x, y) {
            print(c(x, y))
            tables.panel
        }
        image.file <- file.path(system.file(package = "plott"),
            "images", "quadrado.gif")
        tables.panel <- rp.image(tables.panel, image.file, pos = c(25,
            75, 110, 150), id = "topo0", action = click)
        image.file1 <- file.path(system.file(package = "plott"),
            "images", "topo1.gif")
        tables.panel <- rp.image(tables.panel, image.file1, pos = c(0,
            0, 350, 60), id = "topo1", action = click)
        image.file2 <- file.path(system.file(package = "plott"),
            "images", "topo2.gif")
        tables.panel <- rp.image(tables.panel, image.file2, pos = c(145,
            58, 205, 61), id = "topo2", action = click)
        tables.panel <- rp.tkrplot(tables.panel, plot, plotfun =
tables.draw,
            pos = c(0, 250, 700, 525))
        rp.do(tables.panel, tables.draw)
    }
}

ffisher<- function (panel.plot = TRUE)
{
    require(tcltk)
    require(rpanel)
    require(tkrplot)
    tables.draw <- function(tables) {
        with(tables, {
            xobs <- as.numeric(xobs)
            prob <- as.numeric(prob)
            ngrid <- as.numeric(ngrid)
            degf1 <- as.numeric(degf1)
            degf2 <- as.numeric(degf2)
            if (distribution == "F-Fisher") {
                xrange <- c(0.01, 10)
                x <- seq(min(xrange[1], xobs * 1.1), max(xrange[2],
                  xobs * 1.1), length = ngrid)
                dens <- df(x, degf1, degf2)
                ylim <- c(0, 1)
            }
            plot(x, dens, type = "l", ylim = ylim, ylab =
paste("Densidade",
                distribution), col = "magenta")
            abline(h = 0, lty = 3, col = "magenta")
            if (distribution == "F-Fisher")
                title.text <- paste("Distribuição F - Fisher(",
                  degf1, ",", degf2, ")", sep = "")
            title(title.text)
        })
        tables
    }
    tables.redraw <- function(object) {
        rp.tkrreplot(object, plot)
```

```
        object
    }
    panel.name <- rp.panelname()
    if (panel.plot && require(tkrplot)) {
        tables.panel <- rp.control("Distribuições", size = c(350,
            600), realname = panel.name, xobs = 1, prob = 0.05,
            distribution = "F-Fisher", degf1 = 1, degf2 = 1,
            ngrid = 100)
        title.ngrid <- "Total de Amostras"
        title.defg1 <- "GL-Numerador"
        title.defg2 <- "GL-Denominador"
        tables.panel <- rp.textentry(tables.panel, ngrid, tables.redraw,
            title = (paste(title.ngrid)), pos = c(140, 135, 170,
                25))
        tables.panel <- rp.textentry(tables.panel, degf1, tables.redraw,
            title = (paste(title.defg1)), pos = c(140, 165, 170,
                25))
        tables.panel <- rp.textentry(tables.panel, degf2, tables.redraw,
            title = (paste(title.defg2)), pos = c(140, 195, 170,
                25))
        click <- function(tables.panel, x, y) {
            print(c(x, y))
            tables.panel
        }
        image.file <- file.path(system.file(package = "plott"),
            "images", "fisher.gif")
        tables.panel <- rp.image(tables.panel, image.file, pos = c(25,
            75, 110, 150), id = "topo0", action = click)
        image.file <- file.path(system.file(package = "plott"),
            "images", "topo1.gif")
        tables.panel <- rp.image(tables.panel, image.file, pos = c(0,
            0, 350, 60), id = "topo1", action = click)
        image.file <- file.path(system.file(package = "plott"),
            "images", "topo2.gif")
        tables.panel <- rp.image(tables.panel, image.file, pos = c(145,
            58, 205, 61), id = "topo2", action = click)
        tables.panel <- rp.tkrplot(tables.panel, plot, plotfun =
tables.draw,
            pos = c(0, 250, 700, 525))
    }
    else {
        if (panel.plot)
            rp.messagebox("O pacote TKRPLOT não está instalado! Este
aplicativo abrirá em 2 janelas separadas.")
        tables.panel <- rp.control("Distribuições", size = c(350,
            250), realname = panel.name, xobs = 1, prob = 0.05,
            distribution = "F-Fisher", degf1 = 1, degf2 = 1,
            ngrid = 100)
        title.ngrid <- "Total de Amostras"
        title.defg1 <- "GL-Numerador"
        title.defg2 <- "GL-Denominador"
        tables.panel <- rp.textentry(tables.panel, ngrid, tables.draw,
            title = (paste(title.ngrid)), pos = c(140, 135, 170,
                25))
        tables.panel <- rp.textentry(tables.panel, degf1, tables.draw,
            title = (paste(title.defg1)), pos = c(140, 165, 170,
                25))
        tables.panel <- rp.textentry(tables.panel, degf2, tables.draw,
            title = (paste(title.defg2)), pos = c(140, 195, 170,
                25))
        click <- function(tables.panel, x, y) {
```

```
            print(c(x, y))
            tables.panel
        }
        image.file <- file.path(system.file(package = "plott"),
            "images", "fisher.gif")
        tables.panel <- rp.image(tables.panel, image.file, pos = c(25,
            75, 110, 150), id = "topo0", action = click)
        image.file1 <- file.path(system.file(package = "plott"),
            "images", "topo1.gif")
        tables.panel <- rp.image(tables.panel, image.file1, pos = c(0,
            0, 350, 60), id = "topo1", action = click)
        image.file2 <- file.path(system.file(package = "plott"),
            "images", "topo2.gif")
        tables.panel <- rp.image(tables.panel, image.file2, pos = c(145,
            58, 205, 61), id = "topo2", action = click)
        tables.panel <- rp.tkrplot(tables.panel, plot, plotfun =
tables.draw,
            pos = c(0, 250, 700, 525))
        rp.do(tables.panel, tables.draw)
    }
}

exponencial<- function (panel.plot = TRUE)
{
    require(tcltk)
    require(rpanel)
    require(tkrplot)
    tables.draw <- function(tables) {
        with(tables, {
            xobs <- as.numeric(xobs)
            prob <- as.numeric(prob)
            ngrid <- as.numeric(ngrid)
            degf1 <- as.numeric(degf1)
            degf2 <- as.numeric(degf2)
            if (distribution == "Exponencial") {
                xrange <- c(0.01, 10)
                x <- seq(min(xrange[1], xobs * 1.1), max(xrange[2],
                  xobs * 1.1), length = ngrid)
                dens <- dexp(x, degf1)
                ylim <- c(0, 1)
            }
            plot(x, dens, type = "l", ylim = ylim, ylab =
paste("Densidade",
                distribution), col = "red")
            abline(h = 0, lty = 3, col = "red")
            if (distribution == "Exponencial")
                title.text <- paste("Distribuição Exponencial(",
                  degf1, ")", sep = "")
            title(title.text)
        })
        tables
    }
    tables.redraw <- function(object) {
        rp.tkrreplot(object, plot)
        object
    }
    panel.name <- rp.panelname()
    if (panel.plot && require(tkrplot)) {
        tables.panel <- rp.control("Distribuições", size = c(350,
            600), realname = panel.name, xobs = 1, prob = 0.05,
            distribution = "Exponencial", degf1 = 1, degf2 = 1,
```

```
            ngrid = 100)
        title.ngrid <- "Total de Amostras"
        title.defg1 <- "Taxa"
        tables.panel <- rp.textentry(tables.panel, ngrid, tables.redraw,
            title = (paste(title.ngrid)), pos = c(140, 135, 170,
                25))
        tables.panel <- rp.textentry(tables.panel, degf1, tables.redraw,
            title = (paste(title.defg1)), pos = c(140, 165, 170,
                25))
        click <- function(tables.panel, x, y) {
            print(c(x, y))
            tables.panel
        }
        image.file <- file.path(system.file(package = "plott"),
            "images", "exponencial.gif")
        tables.panel <- rp.image(tables.panel, image.file, pos = c(25,
            75, 110, 150), id = "topo0", action = click)
        image.file <- file.path(system.file(package = "plott"),
            "images", "topo1.gif")
        tables.panel <- rp.image(tables.panel, image.file, pos = c(0,
            0, 350, 60), id = "topo1", action = click)
        image.file <- file.path(system.file(package = "plott"),
            "images", "topo2.gif")
        tables.panel <- rp.image(tables.panel, image.file, pos = c(145,
            58, 205, 61), id = "topo2", action = click)
        tables.panel <- rp.tkrplot(tables.panel, plot, plotfun =
tables.draw,
            pos = c(0, 250, 700, 525))
    }
    else {
        if (panel.plot)
            rp.messagebox("O pacote TKRPLOT não está instalado! Este
aplicativo abrirá em 2 janelas separadas.")
        tables.panel <- rp.control("Distribuições", size = c(350,
            250), realname = panel.name, xobs = 1, prob = 0.05,
            distribution = "Exponencial", degf1 = 1, degf2 = 1,
            ngrid = 100)
        title.ngrid <- "Total de Amostras"
        title.defg1 <- "Taxa"
        tables.panel <- rp.textentry(tables.panel, ngrid, tables.draw,
            title = (paste(title.ngrid)), pos = c(140, 135, 170,
                25))
        tables.panel <- rp.textentry(tables.panel, degf1, tables.draw,
            title = (paste(title.defg1)), pos = c(140, 165, 170,
                25))
        click <- function(tables.panel, x, y) {
            print(c(x, y))
            tables.panel
        }
        image.file <- file.path(system.file(package = "plott"),
            "images", "exponencial.gif")
        tables.panel <- rp.image(tables.panel, image.file, pos = c(25,
            75, 110, 150), id = "topo0", action = click)
        image.file1 <- file.path(system.file(package = "plott"),
            "images", "topo1.gif")
        tables.panel <- rp.image(tables.panel, image.file1, pos = c(0,
            0, 350, 60), id = "topo1", action = click)
        image.file2 <- file.path(system.file(package = "plott"),
            "images", "topo2.gif")
        tables.panel <- rp.image(tables.panel, image.file2, pos = c(145,
            58, 205, 61), id = "topo2", action = click)
```

```
        tables.panel <- rp.tkrplot(tables.panel, plot, plotfun =
tables.draw,
            pos = c(0, 250, 700, 525))
        rp.do(tables.panel, tables.draw)
    }
}

gama<- function (panel.plot = TRUE)
{
    require(tcltk)
    require(rpanel)
    require(tkrplot)
    tables.draw <- function(tables) {
        with(tables, {
            xobs <- as.numeric(xobs)
            prob <- as.numeric(prob)
            ngrid <- as.numeric(ngrid)
            degf1 <- as.numeric(degf1)
            degf2 <- as.numeric(degf2)
            if (distribution == "Gama") {
                xrange <- c(0.01, 10)
                x <- seq(min(xrange[1], xobs * 1.1), max(xrange[2],
                  xobs * 1.1), length = ngrid)
                dens <- dgamma(x, degf1, degf2)
                ylim <- c(0, 1)
            }
            plot(x, dens, type = "l", ylim = ylim, ylab =
paste("Densidade",
                distribution), col = "purple")
            abline(h = 0, lty = 3, col = "purple")
            if (distribution == "Gama")
                title.text <- paste("Distribuição Gama(", degf1,
                  ",", degf2, ")", sep = "")
            title(title.text)
        })
        tables
    }
    tables.redraw <- function(object) {
        rp.tkrreplot(object, plot)
        object
    }
    panel.name <- rp.panelname()
    if (panel.plot && require(tkrplot)) {
        tables.panel <- rp.control("Distribuições", size = c(350,
            600), realname = panel.name, xobs = 1, prob = 0.05,
            distribution = "Gama", degf1 = 1, degf2 = 1, ngrid = 100)
        title.ngrid <- "Total de Amostras"
        title.defg1 <- "Forma"
        title.defg2 <- "Taxa"
        tables.panel <- rp.textentry(tables.panel, ngrid, tables.redraw,
            title = (paste(title.ngrid)), pos = c(140, 135, 170,
                25))
        tables.panel <- rp.textentry(tables.panel, degf1, tables.redraw,
            title = (paste(title.defg1)), pos = c(140, 165, 170,
                25))
        tables.panel <- rp.textentry(tables.panel, degf2, tables.redraw,
            title = (paste(title.defg2)), pos = c(140, 195, 170,
                25))
        click <- function(tables.panel, x, y) {
            print(c(x, y))
            tables.panel
```

```
        }
        image.file <- file.path(system.file(package = "plott"),
            "images", "gama.gif")
        tables.panel <- rp.image(tables.panel, image.file, pos = c(25,
            75, 110, 150), id = "topo0", action = click)
        image.file <- file.path(system.file(package = "plott"),
            "images", "topo1.gif")
        tables.panel <- rp.image(tables.panel, image.file, pos = c(0,
            0, 350, 60), id = "topo1", action = click)
        image.file <- file.path(system.file(package = "plott"),
            "images", "topo2.gif")
        tables.panel <- rp.image(tables.panel, image.file, pos = c(145,
            58, 205, 61), id = "topo2", action = click)
        tables.panel <- rp.tkrplot(tables.panel, plot, plotfun =
tables.draw,
            pos = c(0, 250, 700, 525))
    }
    else {
        if (panel.plot)
            rp.messagebox("O pacote TKRPLOT não está instalado! Este
aplicativo abrirá em 2 janelas separadas.")
        tables.panel <- rp.control("Distribuições", size = c(350,
            250), realname = panel.name, xobs = 1, prob = 0.05,
            distribution = "Gama", degf1 = 1, degf2 = 1, ngrid = 100)
        title.ngrid <- "Total de Amostras"
        title.defg1 <- "Forma"
        title.defg2 <- "Taxa"
        tables.panel <- rp.textentry(tables.panel, ngrid, tables.draw,
            title = (paste(title.ngrid)), pos = c(140, 135, 170,
                25))
        tables.panel <- rp.textentry(tables.panel, degf1, tables.draw,
            title = (paste(title.defg1)), pos = c(140, 165, 170,
                25))
        tables.panel <- rp.textentry(tables.panel, degf2, tables.draw,
            title = (paste(title.defg2)), pos = c(140, 195, 170,
                25))
        click <- function(tables.panel, x, y) {
            print(c(x, y))
            tables.panel
        }
        image.file <- file.path(system.file(package = "plott"),
            "images", "gama.gif")
        tables.panel <- rp.image(tables.panel, image.file, pos = c(25,
            75, 110, 150), id = "topo0", action = click)
        image.file1 <- file.path(system.file(package = "plott"),
            "images", "topo1.gif")
        tables.panel <- rp.image(tables.panel, image.file1, pos = c(0,
            0, 350, 60), id = "topo1", action = click)
        image.file2 <- file.path(system.file(package = "plott"),
            "images", "topo2.gif")
        tables.panel <- rp.image(tables.panel, image.file2, pos = c(145,
            58, 205, 61), id = "topo2", action = click)
        tables.panel <- rp.tkrplot(tables.panel, plot, plotfun =
tables.draw,
            pos = c(0, 250, 700, 525))
        rp.do(tables.panel, tables.draw)
    }
}
```